.EXE

MARCH 1991 VOL 5 ISSUE 9

The Software Developers' Magazine

Libraries, pros and cons. We polish the arguments.

How to wrap up a C library interface in a shining, modern OOP cocoon.

We have a brand new routine for our Super-VGA graphics collection.

How the Germans do their hacking.

Borland breaks the SDK barrier - There's a new C++ for Windows.

THE TRAINING LEADERS

To 'C' PROGRAMMERS AT ALL LEVELS

"The best path for complete 'C' training begins at QA".

'C' has become the dominant language in computer programming and provides access to the software industries major application program interfaces.

'C' is probably now, or shortly will be, your way forward in program development – and QA can really help you along that path.

Only QA's extensive 'C' curriculum totally supports the training needs of all 'C' programmers, from beginner through to highly experienced professional. Each course can stand alone or can be entry point to reach any level of skill or knowledge you may set yourself.

In common with all QA courses, our instructors are experienced professionals themselves, our training materials are perfected for efficient learning, and all our own skill and energy is channelled toward helping you get the maximum benefit from your training.

Send for more details today or call Samantha Trinder on 0285 655888.

BEGINNERS START HERE

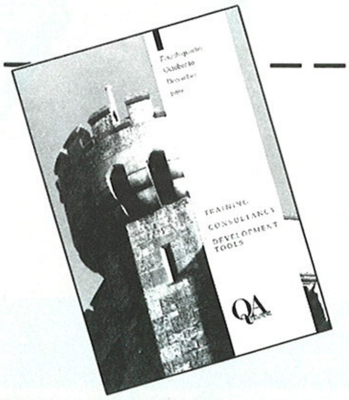
OTHERS START HERE...

...OR ANYWHERE HERE...

...INCLUDING C++

'C' COURSE OPTIONS

C PRIMER 2 Days £340	HLL concepts via C • Overview of C programming environment • Developing simple C programs • Data typing, block structure, scope concepts • Introducing C functions and modular programming
C PROGRAMMING 4 Days £720	C programming structure and style • Data types and variables • Flow of control in C programs • Pointers and address manipulation • Portability and efficiency considerations • Debugging C programs • Working with larger C programs
C TECHNIQUES FOR OS/2 & P.M. 1 Day £180	Using C in an OS/2 environment • PC and OS/2 extensions to C • Mixed Models of compilation • Advanced C techniques for OS/2 • The OS/2 API C interface • Dynamic link library concepts • OS/2 development tools overview
C TECHNIQUES FOR DOS 1 Day £180	Implications of using C under DOS • PC and DOS extensions to C • Models of compilation and segmentation • Interfacing to DOS and BIOS from C • Linking C to other languages • DOS development tools overview
C ADVANCED SOFTWARE DEVELOPMENT 4 Days £760	A lightning tour of C • Advanced pointer techniques • 'Defensive' programming techniques in C • Advanced C concepts • A closer look at C's library routines • Recent ANSI and SAA C developments • C development tools overview
C STANDARDS 1 Day £220	C standards – aims, goals and benefits • Defining practical and realistic standards • Naming and layout standards • C program organisation standards • Defensive C programming • Portability, maintainability and efficiency
C++ PROGRAMMING 4 Days £780	Object oriented programming concepts • OOP and C++ • Concept of classes • Class hierarchys, derivation and inheritance • Constructors and Destructors • Application examples
OBJECT ORIENTED DESIGN USING C++ 3 Days £680	OOP and OOD: the need • OOD: fundamental concepts • Identifying objects and relationships • Object properties • Stages in object oriented design • C++ design guidelines • Development team roles • Tooling up for OOD



Please complete and mail to QA Training, Cecily Hill Castle, Cirencester, Glos GL7 2EF.

CIRCLE NO. 550

☐ Tick here for full details on all our 'C' courses.

Send me details on (tick as appropriate):

☐ C Primer ☐ C Programming ☐ C Techniques for OS/2 & PM
☐ C Techniques for DOS ☐ C Advanced Software Development
☐ C + + ☐ OS/2 Courses ☐ Windows Courses ☐ Support Courses

NAME ____ TITLE ____

COMPANY ____

ADDRESS ____

TEL. NO. ____

QA TRAINING

Window

Help

Vol 5 Issue 9

.EXE

March 1991

General

.EXE Magazine is independent and not affiliated to any vendor of hardware, software or services. It is published by Process Communications Ltd, 10 Barley Mow Passage, Chiswick, London W4 4PH.
Tel: (Advertising/Editorial/Production) 081 994 6477
(Subscriptions) 044282 4501
Facsimile: 081 994 1533 Telex: 8811418 SPACES G
ISSN: 0268-6872

Subscriptions

.EXE Magazine is a monthly journal for software developers. It is available only by subscription, at a cost of £35 per annum (11 issues) in the UK (see insert card for academic and overseas rate). 'A subscription implies that this journal will be sent to the subscriber until one of the three expires' - AG Macdonell. The magazine is published around the 26th of the month preceding the cover date. There is no January issue. To subscribe, please call 044282 4501, or write to The Subscriptions Manager, .EXE Magazine, 10 Barley Mow Passage, Chiswick, London W4 4PH. We can invoice your company or take ACCESS and VISA cards. In case of a query about your subscription, please call 044282 4501.
Back issues are available at a cost of £3.50 each - call our Chiswick office (081 994 6477) for a list of issues still in print. We also offer a range of magazine binders, disk holders etc - please call for details.

Editorial

Editorial enquiries should be addressed to The Editor, .EXE Magazine, 10 Barley Mow Passage, Chiswick, London W4 4PH. We welcome letters, opinions, suggestions and articles from our readers. If you are interested in contributing articles, please write to this office for a copy of our Contributors' Guide.
Information contained in .EXE is believed to be correct. If errors are found, we will endeavour to publish a clarification in the next available issue.
From time to time, we offer to copy the PC program code described in an article onto diskette for our readers. In this case, please send a blank, formatted disk with a self-addressed, prepaid mailer to the editorial address given above. We can copy both 5.25" and 3.5" disks.
The publishers can accept no liability for any consequences of using software distributed in this way.

Advertising

If you are interested in advertising in this magazine, please write to the Display Advertising Manager/Recruitment Sales Manager (as appropriate), .EXE Magazine, 10 Barley Mow Passage, Chiswick, London W4 4PH, or call 081 994 6477 for details of our advertising rates.

Pronunciation

The name of .EXE Magazine is pronounced to rhyme with 'not sexy magazine'.

Copyright

Special Issue - Libraries

Casting the net

Many office workers could stay at home, operating through a computer, fax and modem. Jules May wonders why so few do so.

While the rents in London sky-rocket, while transport in our cities becomes constipated, and now petrol supplies are drying up, telecommuting is hailed as 'exactly what we need to solve the problem'. When GUIs first appeared, when virtual realities emerged from the research labs, even when data telecommunications started to be implemented, telecommuting was claimed to be 'practical at last'. It seems telecommuting is wheeled out in response to almost every crisis, and also in response to every technical development.

It is now possible to collect all the information that is required to complete a task, perform it, and deliver the results to where they are supposed to be, entirely electronically. All parts of the process can be performed more efficiently than in traditional practices.

The advantages are obvious. Without having to travel to and from the office, the working day is shorter. Workers can build an environment which is just right for them, rather than in what the employer thinks will give a good impression. There are direct gains for the employer, too. For every employee who works at home, there is one fewer desk to provide, less office to clean, heat, light and so on. Overall, less fuel is burned ferrying people about, which is good for everybody. Above all, people are more productive at home than in their office-based counterparts - according to a conference organised by British Telecom and the CBI, up to four times as productive!

Of course, working from home doesn't suit everybody. Certain workers have to be at a certain place on time - workers who deal with the public, such as receptionists and salesmen, clearly must travel to where the people are. Workers who actually make things need access to machinery and raw materials, and these are better handled at a central point. Even workers who deal with information may not be suited to telecommuting; some people find it hard to work at home. These disadvantages aside, surveys suggest that most people doing work appropriate for telecommuting would prefer to work from home, even with a drop in salary equivalent to the savings on transport, and some sources suggest that as many as 80% of all workers are doing appropriate work.

And yet, very few people actually work this way. In a recent conversation on CIX between contract workers (surely one of the best places to find such people), only four of us admitted to telecommuting. In another conference, dealing specifically with telecommuting, words such as 'dream' are used with disarming regularity. One contributor explained how his contract had been modified to preclude such working when he mentioned it to his employer!

So, why are so few people working this way? The answer lies with our management. If people are working at home, how do you know that the worker is really doing all he is paid for. What kind of conditions does one set?

Any company, to have any sense of where it is going, must have some objectives - that is, what they want to achieve, by when, and how much it ought to cost. Why not give the workers (who will implement the plan) precisely those objectives and precisely those deadlines? It happens now - 'I want this report in time for tomorrow's meeting', or 'This piece of code must be ready for testing in two months'. If the worker can meet these goals doing only two hours work daily at home (instead of eight hours in an office), why should the employer object?

The fact is, all my experience has convinced me that the majority of employers in this country don't give a damn about productivity. They have bought five days per week of that individual's time, and if he spends that time competing in internal politics, chatting over coffee breaks, playing practical jokes and gazing out of the window, that is fine so long as he doesn't take advantage of the employer. If this were not so, why are programmers paid by the hour instead of by milestones, and how many programmers can charge their employers for time spent writing down solutions that occurred to them in front of the TV?

Homeworking, in all its forms, is bound to come - the advantages are too great for it to be otherwise. The question is, when will it come, and how painful will it be? With countries, and companies, putting up satellites for just that purpose, are we going to take advantage, or are we going to wait, yet again, until the rest of Europe have beaten us to it?

EXE

This article was prepared at home, and submitted to the .EXE office via CIX. The author can be contacted as jules@cix.compulink.co.uk, or on 0707 44185.

C LIBRARIES

DATABASE/FILEHANDLING

Btrieve for OS/2	OS/2	£330
Btrieve for Windows	MS-DOS	£330
Btrieve v5.1	MS-DOS	£330
XQL	OS/2,PC-DOS	£440
Xtrieve Plus	OS/2,PC-DOS	£330
C/Database T'lchest sce	PC-DOS	£ 39
CBTREE	(source any C)	£135
C-Index/Plus	(source any C)	£175
C-ISAM (L,MS)	MS-DOS	£215
C-tree	(source any C)	£210
C-tree Plus	PC-DOS	£350
Essential Btree (s'ce)	PC-DOS	£120
CQL s'ce	PC-DOS	£270
Faircom Toolbox Prof	PC-DOS	£610
Pro-C & Workbench v2	PC-DOS	£585
R-tree	MS-DOS	£160
db-VISTA DBMS Object	PC-DOS	£450
Lattice dBCIII (L,MS)	MS-DOS	£185
Lattice dBCIII+	OS/2&MS-DOS	£365
SftFcus Btree&Isam	(sce any C)	£ 80

GRAPHICS

Enhanced Graphics Tlkt	PC-DOS	£170
Essential Graphics v3	PC-DOS	£215
GFX Graphics	PC-DOS	£ 80
GraphiC v5 (CI,L,DS,MS)	PC-DOS	£290
graphics MENU/META v3	PC-DOS	£135
GSS Kernel system	OS2,PC-DOS	£425
GSS Graph Dev T'kit	OS2,PC-DOS	£425
HALO v3 (MSC5,LAT,TC)	PC-DOS	£210
HALO Professional (C)	PC-DOS	£335
HGraph (MSC)	PC-DOS	£ 80
Ingraf (MSC)	PC-DOS	£185
MetaWINDOW v3.7b	PC-DOS	£150
MetaWINDOW/Plus	PC-DOS	£200
Professional Halo	PC-DOS	£335
TurboWINDOW/C (Turbo)	PC-DOS	£ 85
QuickWINDOW/C (QC)	PC-DOS	£ 85

SCREEN & WINDOWS

C-Scape +Look&Feel	OS/2&PC-DOS	£345
Vitamin C for OS/2	OS/2&DOS	£240
Blaise View Mngr.(s'ce)	PC-DOS	£245
Curses/PC w. s'ce	PC-DOS	£240
Entelekon Windows(s'ce)	PC-DOS	£105
Greenleaf Datawindow	PC-DOS	£210
Greenleaf Makeform	PC-DOS	£ 70
Lattice Curses	PC-DOS	£105
Multi-windows (MS,L)	PC-DOS	£190
Panel Plus II (source)	PC-DOS	£270
Power Screen (MS&TC)	PC-DOS	£105
Vermont Views (MSC,TC)	PC-DOS	£345
Vitamin C (source)	PC-DOS	£145

GENERAL FUNCTIONS

C/Utilities T'chest sce	PC-DOS	£ 39
C-Worthy v1.2 + forms	MS-DOS	£340
C Tools Plus/6.0 (MS)	PC-DOS	£ 95
CodeRunner	PC-DOS	£105
Entelekon Funct(source)	PC-DOS	£105
Essential C Utils v5	PC-DOS	£145
G'leaf Fin'cial Mathlib	PC-DOS	£210
G'leaf Super Functions	PC-DOS	£160
Greenleaf Functions s'ce	PC-DOS	£125
Lattice C Comp Kit+s'ce	PC-DOS	£160
PDK£1	PC-DOS	£ 75
Power Search	PC-DOS	£ 95
Resident-C (MS,L)(s'ce)	PC-DOS	£145
Silverware C EMM Lib	PC-DOS	£115
The Heap Expander(MS&TC)	PC-DOS	£ 65
Turbo C Tools v2	PC-DOS	£ 95
WKS Library	PC-DOS	£160

COMMS LIBRARIES

C Asynch Manager V3s'ce	PC-DOS	£120
Essential Comms(L,MS,T)	PC-DOS	£175
Greenleaf Comms(source)	PC-DOS	£195
Multi Comms (L,MS)	PC-DOS	£190
Silvercomm C Asynch	PC-DOS	£145

SCIENTIFIC LIBRARIES

C Lang Sci Lib v2	ANY C	£160
Mathpak 87 (L,MS)	MS-DOS	£ 80
Science & Eng Tools	MS-DOS	£ 59

PROGRAMMERS UTILITIES

PC-Lint	OS/2,MS-DOS	£ 85
C-Documentor	PC-DOS	£195
C-Scan	PC-DOS	£195
Clear+ for C	PC-DOS	£140
Lattice Comp Companion	MS-DOS	£ 65

REAL TIME & MULTI-TASKING TOOLS

Concurrent C (PC/MPX)sce	MS-DOS	£ 60
Multi-C (L,MS,CI,TC)	PC-DOS	£190
Op.Sys.T'box (MS) s'ce	PC-DOS	£ 69
Timeslicer v6 (MS6)	PC-DOS	£200
Over-C (L,MS)	PC-DOS	£225

C++ COMPILERS

Guidelines C++	PC-DOS	£370
Turbo C++	PC-DOS	£120
Turbo C++ Professional	PC-DOS	£200
Zortech C++	PC-DOS	£120
Zortech C++ Dev Edition	PC-DOS	£270
Zortech C++ OS/2 Dev Ed	OS/2	£360
Zortech C++ 386 Dev Ed	PC-DOS	£540

LIBRARIES & SUNDRIES

C++/Views (ZC++)	PC-DOS	£270
Greenleaf Comms (T&ZC++)	PC-DOS	£195
Object Vision	PC-DOS	£325
Zinc Interface Lib (TCH)	PC-DOS	£150
Zortech C++ Database	PC-DOS	£180
Zortech C++ Video Course		£270

PROGRAMMING TOOLS

Ada Compilers	Algol Compilers
Assemblers & Libs	AWK
Basic Compilers	Basic Interpreters
Basic Utilities	Basic Libraries
BCPL Compilers	C Compilers
C Interpreters	C Libraries
C Utilities	Cobol Compilers
Comms.Libraries	Cross Assemblers
Database Libs.	Debuggers
Dis-assemblers	Editors
Engineers Libs.	Expert Systems
Forth	Fortran Compilers
Fortran Libraries	Graphics Libraries
Icon	Linkers/Locaters
Lisp	Modula-2
Nial Interpreters	OPS 5
Pascal Compilers	Pascal Libraries
Prolog	Rexx
Risc	Screen Libraries
Smalltalk	Snobol

We stock many items for which there is no space in these advertisements.

DISK COPYING SERVICE

We can copy files to and from 600 disk formats including CP/M, CP/M-86, MS-DOS, PC-DOS, ISIS, APPLE, SIRIUS, BBC, TORCH, APRICOT, HP-150, TRSDOS, DEC RT-11, IBM BEF, ATARI ST, AMSTRAD, MACINTOSH.

Our charge is £10.00 + disk + VAT with discounts on small quantities and disks are normally despatched within 24hrs of receipt. For more information call us.

PROGRAM EDITORS

BRIEF v3 for OS/2&DOS	OS/2&DOS	£195
Kedit v4 for OS/2	OS/2&PC-DOS	£125
Epsilon v5 for OS/2	OS/2	£135
QEdit Advanced for OS/2	OS/2	£ 55
MKS Vi for OS/2	OS/2	£130
Sage for OS/2&DOS	OS/2&PC-DOS	£180
EC Editor v2.1	PC-DOS	£ 40
Epsilon v5	PC-DOS	£135
Kedit v4	PC-DOS	£105
ME v2.01 (w.C source)	PC-DOS	£155
MIX Editor	MS-DOS	£ 20
MKS VI	PC-DOS	£ 99
Multi-Edit v5	PC-DOS	£ 69
Multi-Edit Professional	PC-DOS	£115
Norton Editor v1.3	PC-DOS	£ 42
QEdit Advanced	PC-DOS	£ 55
RED v6.6 (with C S'ce)	PC-DOS	£ 65
Syndie	MS-DOS	£445
XTC (with Pascal source)	PC-DOS	£ 50
Vedit-Plus v3	PC-DOS	£120
Vedit-Plus (CRT)	MS-DOS	£120
Vedit-Plus	CP/M-86	£120
MIX Editor	Z80 + CP/M-80	£ 25
Vedit-Plus v2.33	CP/M-80	£110

For more information please call us.

THE C LANGUAGE

Microsoft C V6 is a complete rewrite with improved optimisation and a new Programmer's Workbench

C COMPILERS

Zortech C++	OS/2	£ 36
Lattice C v6	OS/2&MS-DOS	£175
Microsoft C v6	OS/2&PC-DOS	£269
Topspeed C for OS/2	OS/2	£335
Watcom C v8 Prof	OS/2&DOS	£335
Avocet AVCOM 86C	PC-DOS	£210
Aztec C86 Professional	MS-DOS	£120
Aztec C86 Developer	MS-DOS	£175
Aztec C86 Commercial	MS-DOS	£265
CI C86 Plus v1.3	MS-DOS	£290
ECO-C88 V4.0	MS-DOS	£ 70
HIGH C (Metaware) v1.7	MS-DOS	£380
HIGH C 386 v2.3	PL386&MS-DOS	£640
Hi-Tech C	MS-DOS	£135
Let's C v4	MS-DOS	£ 60
Microsoft QuickC v2.5	PC-DOS	£ 60
M'soft QuickC/QuickAsm	PC-DOS	£105
MIX C	MS-DOS	£ 20
MIX Power C & lib s'ce	PC-DOS	£ 29
Turbo C++	PC-DOS	£120
Topspeed C Standard	PC-DOS	£135
Topspeed C Extended	PC-DOS	£270
Watcom C v8 std	MS-DOS	£269
Watcom C/386 std	PL386&MS-DOS	£625
Watcom C/386 Prof	PL386&MS-DOS	£830
Zortech C++ v2.1	PC-DOS	£120
Zortech C++ Dev Ed	PL386&DOS	£540
Aztec C86 Developer	CP/M-86	£245
Hi-Tech C	CP/M-86	£150
Lattice C	CDOS	£380
Avocet C	CP/M-80	£210
Aztec C Personal 1.06D	CP/M-80	£120
Aztec Commercial 1.06D	CP/M-80	£200
Hi-Tech C	Z80 + CP/M-80	£ 99
Mix C	Z80 + CP/M-80	£ 35
Aztec C65 V1.05	Apple DOS	£200
Aztec C68K MPW C	MACINTOSH	£110
Aztec C68K C	MACINTOSH	£ 75
Aztec C68K/AM Developer	AMIGA	£175
Aztec C68K/AM Prof'snl	AMIGA	£120
Aztec C68K/ST Developer	ATARI	£175
Aztec C68K/ST Prof'snl	ATARI	£120
Hi-Tech C	ATARI	£ 99
Mark Williams C v2	ATARI	£110
Laser C	ATARI	£135
Prospero C	ATARI	£105

C CROSS COMPILERS

We supply 2500AD, Avocet, Aztec, Lattice, IAR and Hi-Tech Cross Compilers hosted on MS-DOS and targeted on Z80, 6502, 6801, 68HC11, 6301, 6809, 7811, 8051, 8096, 68000 & 68020. Please call for information or advice.

C INTERPRETERS

Instant C/16M supports large programs on 286/386 machines.

C-terp V3.0X	PC-DOS	£165
Instant C/16M	PC-DOS	£700
Interactive C	PC-DOS	£195
Introducing C	PC-DOS	£ 85
Living C Plus	PC-DOS	£135
RUN/C Professional	MS-DOS	£110

PROLOG LANGUAGE

Arity/Prolog Comp	OS/2	£745
ADA Educ.Prolog	MS-DOS	£ 26
ADA FS Prolog	MS-DOS	£ 45
ADA VMI Prolog	MS-DOS	£ 69
ADA VML Prolog	MS-DOS	£150
Arity/Prolog Comp. v6	PC-DOS	£520
Arity/Prolog Int. v6	PC-DOS	£220
Arity Standard	PC-DOS	£ 75
Flex Expert Dev Sys	PC-DOS	£1210
LPA PROLOG Prof v3	PC-DOS	£755
Micro-PROLOG v3.1	MS-DOS	£ 70
PDC Prolog	PC-DOS	£240
Micro-Prolog v3.1	CP/M-86	£ 70
Micro-Prolog v3.1	CP/M-80	£ 70
Prolog-1 v2.2	CP/M-80	£225
LPA Mac Prolog v3	Apple Mac Pls	£450

PRICES & DELIVERY

Prices do not include VAT or other local taxes but do include delivery in UK and Europe. Please check prices at time of order, ads are prepared some weeks before publication.

This page lists some of our products. Call us for a complete pricelist.

Order by phone with your credit card.

GREY MATTER

2 Prigg Meadow, Ashburton, Devon TQ13 7DF

TEL. (0364) 53499

GREY MATTER

2 Prigg Meadow, Ashburton, Devon TQ13 7DF

TEL. (0364) 53499

GREY MATTER

2 Prigg Meadow, Ashburton, Devon TQ13 7DF

TEL. (0364) 53499

CIRCLE NO. 551

Pain-free copy
Disk Duplicator Plus is a clever little DOS program that performs intelligent copying of floppy disks (not duplicating empty tracks, buffering to hard disk etc). Version 3.0 has just come out, which can exploit expanded and extended memory and has a new 'batch mode' for copying multiple source disks. It costs £49.95 from Performance Technologies Ltd (0344 488118).

DOS 386 Professional
DOS 386 Professional is a multi-tasking/multi-user OS like Concurrent DOS; the difference is that it also offers networking and distributed processing. A complete system is made up of a fair hotch-potch of hardware: PC-based servers, PC workstations and good old-fashioned character terminals. Applications can be directed to run on any server/workstation in the system, and each user can control up to 4 multitasking sessions. UK manufacturer IMS claims the system can support over 1000 users. Starting price is £1195,IMS is on 0276 686569.

Sidekick 2.0
You remember Sidekick? It was innovative and slick. You remember Sidekick Plus? It was fat and over-complex. Borland has now launched Sidekick 2.0, which is an attempt to consolidate Sidekick's neatness with Sidekick Plus's trillion features. Thus SKP's outliner, file manager and 'Invaders' game have been dropped. The Time Planner, Address Book and Communications applications produce Paradox data files, and a new printer driver has been added which offers Postscript support. Thinking to try out this last feature, I attempted to print a notepad text file - but I had set up the wrong printer port. Sidekick 2.0 went into an endless, unbreakable loop. Hmm.

C-Terp gets extended
C-terp, the Gimpel C interpreter, has now reached V3.5. The system now runs in protected mode, c/o Phar Lap, and the editor has been improved to include multiple undo. Better external linkage, too. Gimpel is on 0101 215 584-4261.

UI widens open systems
Unix, the AT&T-led open systems consortium has laid out its Roadmap plans for this year. They include a UNIX API for parallel processing, distributed computing, and support for Open Systems Interconnect (OSI) technology. Details from Unix International Belgium - 010 322 6723700.

And now, here is next year's news...

Microsoft has announced many of its plans for the short and medium term. During the recent Windows Show at Olympia, Microsoft UK's Andrew King briefed .EXE on what we can expect from the company in future months. There's a lot, so pay attention.

Operating systems: as has been widely reported, DOS 5.0 is now in late beta-test (more than 7000 beta sites, product will be 'rock solid' says Microsoft): MS-DOS 5.0 will play DR DOS-style high memory tricks, to maximise the amount of space available within the 640 KB. QuickBASIC will be bundled with the package, plus a full screen editor. It will have task-switching (supported by a new API) and ordinary DOS programs will have the ability to access extended memory via a DPMI interface. You will be able to buy it - or at least upgrade to it - from ordinary retail outlets (abandoning the current policy of forcing you to get it from your hardware supplier), although you will have to hand in an earlier version. DOS 5.0 will be ROMable. As will be Windows 3.1 (ETA: Summertime), in anticipation of a wave of fancy laptops. W3.1 will feature vector fonts, increased prettiness (file manager, program manager), run faster and better networking. The SDK will contain an new DDE manager (as currently featured in OS/2 PM), plus support for OLE - see February's news for a description of this. There will also be a few standard dialogs - to be used by programmers as building blocks - for operations such as File Open. W3.1 will contain the handles the new hardware developments such as pens and multimedia; you will be able to use the pen to control older applications without adjusting the software.

Broadening the outlook, Microsoft is working on a new operating system called New Technology. 99% written in C, NT will be portable across a range of processors, including non-Intel RISC chips - the current prototype runs on 386/486s and an (unspecified brand) RISC. It will also support multi-processing and clever networking (a network redirector). NT will form the basis of OS/2 *V3.0*, which will contain a 32-bit version of Windows, and a POSIX interface as well as the API's for DOS and OS/2 V1.x/V2.x. From this point, all Windows applications will be portable to OS/2 and, provided that they are written in C and can be recompiled, for RISC-based NT platforms. King emphasised that Microsoft remains 100% behind OS/2, and that the love affair with IBM is not over.

Software tools: Microsoft will continue to develop two separate streams of languages: the low end Quick... range and the high end heavyweights. The Quick languages will be ported to Windows (including a Windows-based editor) in the next 12 months. There will be a QuickC++. Also before a year has passed, there will be tools that let you write Windows programs in BASIC, FORTRAN, COBOL and Pascal. The next version of C, which I shall henceforth call C 7.0, will have the following features: a new character-based PWB with increased speed and capacity plus an object-aware browser, a full BSI ANSI validation (a bit presumptuous that; perhaps they should wait until they get the certificate), inline function and inter-module optimisation, a VROOMM-like overlay capability, and possibly (AK became enigmatic at this point) 386/486 code generation.

There's loads more stuff to tell you about, including proposed Microsoft C++ extensions (including exception handling, templates **and** garbage collection), a Windows application framework called AFX, pcode generation in the C compiler... but it will all have to wait, because I'm out of space. Sorry.

Multi-Edit upgrade

American Cybernetics has announced the new version of Multi-Edit, its popular programmers editor. The text editor now offers expanded memory support, and a choice of SAA or classic interfaces. An expanded on-line help and 380 page User's Guide is also provided. Multi-Edit V5.0 is marketed from a number of sources in the UK - phone AC on 0101 602 968 1945 for details.

Clipper conference

If you are a Clipper developer and you missed Nantucket's recent European Developer's Conference, reckon you missed a treat. Many Clipper luminaries were present, including Nantucket chief Larry Heimendinger, who rallied his forces behind the as yet unproven Clipper 5.0. Most impressive were the printed notes given out to accompany the talks. These consisted of proper full-text reference material with code examples, instead of the crummy photostats of the lecturer's overhead projector slides which everybody else in the industry seems to use. We must make the Software Tools '91 documentation like Nantucket's, plug, plug.

Phar Lap release 286 Dos Extender

That's right, for the 80286. The new product will work just as the 386 version, so applications can often take advantage of it by simply relinking their Microsoft C applications. The extender has a number of pleasant features: particularly impressive is the 286 Dos Extender's support for DLLs. And DOS programmers seeking OS/2 compatibility (or perhaps more likely, vice versa) will note that a subset of the character based OS/2 API is included. Also included is a feature to make extended memory available to the Microsoft C compiler itself, allowing the compilation of >640 KB programs without worries. Information is a little scanty at the moment, as the product was just being announced at the Software Developers' Conference in Santa Clara as we went to press. More details when we get them. In the meantime, 286 programmers with deadlines and 'out of memory' errors can phone Phar Lap's dealer in this country, System Star: 0992 500919. The price of the 286 Dos Extender is the same as that of the 386, £315. Run-time royalties will be around £2 per copy.

DOS in real time

This sounds like a sensible development: Intel has produced a version of its RMX real-time operating system which can run MS-DOS as a task. DOS/RMX is based on the company's 32-bit iRMXIII product. To use it, you boot up your 386 machine with an ordinary version of DOS, then execute a special TSR which loads and runs iRMX in high (> 1 MB) memory. DOS continues to execute in a 'virtual 8086', and you can 'hotkey' between DOS and RMX using the SysReq key. You can't run DOS programs which use 286/386 protected modes, but there is a LIM 4.0 driver which can allocate some of RMX's extended memory to DOS. There are facilities for inter-process communication between the DOS and RMX programs (which Intel says is easier to code than Windows' DDE system) and the disk files of both systems can be accessed by applications of the other.

The idea is that you can use MS-DOS applications and tools to create the pretty, front-end part of your real time application, with RMX handling the tricky stuff in the background. At the launch, Intel illustrated this idea with a (real mode) animated Windows 3 application fronting a process control system. DOS/RMX costs £84 for the run-time system and £1633 for the development kit. Intel is on 0793 696000.

TurboPower updates to V6.0

TurboPower is a company which produces a number of handy utilities for Turbo Pascal users. Unfortunately, Borland rewrote the heap manager for Turbo Pascal V6.0 (converting it back to the algorithm used in V3.0) and promptly broke all TurboPower's products. So if you are using Object Professional, B-Tree Filer, Turbo Analyst or Turbo Professional, and are thinking of upgrading to Turbo Pascal V6.0, you'll need to get an ten dollar update pack. On a happier note, the company is now selling the TSR utilities from the Object Professional class library separately. TSR Made Easy costs $49, and provides Turbo Pascal and assembly language routines for TSR swapping to EMS, XMS or disk, selectable hot keys, keystroke playback and ISR handling. The package includes full source, documentation and a number of demo programs. It's compatible with all Turbo Pascals from V5.0 onwards. TurboPower are on 0101 408 438 86008.

Superbase now with SQL

Superbase 4, the popular database for Windows applications, has just been released in V1.3. New developments include the ability to embed SQL commands in any DML (Superbase's development language) context, on-screen refreshing of networked databases and better image handling features. Pre and post processing during data entry (always an attractive feature in these programs) is now also supported. Also announced is the inevitable SQL client link for the package, with support for Microsoft, Sybase and Gupta servers. The new SQL Library costs £295, and can be ordered from Precision on 081 330 7166.

Reading Mac Disks on a PC

...is impossible. Well, it used to be: old Macs varied their head speed while writing to disc, making it physically impossible to grab the data on a standard PC drive. Since January 1989, though, Macs have been sold with a 1.44 MB SuperDrive. This uses the same physical sector layout as the IBM's high density drives, allowing (in theory) some sort of data transfer. Access Mac from Avantek is the software that does the rest - unravelling the Mac's logical format, and transferring the results into MS-DOS files, with their stubby little filenames and dry, upper-case attitudes.

Access Mac costs £113.16, and includes technical support. You can contact Avantek on 031 660 5213.

Sp.

Award for most counter-productive publicity this month goes to Attica Cybernetics, who sent us a screen shot of their new, 25,000 entry CD-ROM encyclopaedia. The accompanying screen shot shows the application proudly running in a little Windows 3 box, together with an application title bar which reads 'Hutchinson Electronic Encycolpedia'[sic].

Paradox Access from 1-2-3

Lotus (0784 455445) is now shipping its DataLens Driver for Paradox Tables. The new driver allows 1-2-3 V3.1 and 1-2-3/G users to directly access information stored in Paradox files using 1-2-3 commands. The driver works with Paradox V2.0 or higher, and is available either directly or from your Lotus Authorised Dealer. It costs £65.

On the cutting edge

MIPS computer systems has unveiled the first 64-bit RISC processor, the R4000. No price was fixed (although they estimate it will cost approximately the same as a 386 chip in mass production), and MIPS remains hesitant about final performance measurements. The pre-announcement was believed to be intended to claim the limelight before similar chips are released this year by Motorola, Texas Instruments and others.

Unix on Transputers

Chorus Systême, the Unix microkernel manufacturer discussed by Peter Collinson in .EXE December, has announced a joint venture with INMOS to port the Chorus kernel to the H1 transputer, to be completed by early 1992. The kernel was chosen because of its modular structure and internal messaging system, which mirrors the transputer's own architecture. The implementation will provide a full Unix V3.2 system, with modules able to be added and removed as required for real-time work. INMOS is on 0454 626616.

Norton Editor V2.0

Symantec, distributors of the Norton range of utilities, has announced a new version of the full screen text editor for IBM compatible PCs. This latest version provides users with unlimited file lengths, and pull-down menus and mouse support. The classic editor is also included, which still runs in 50K of memory. It costs £60, and is available from Symantec or its dealers for £60. Existing users can upgrade to the latest version for £20.
Symantec are on (0628) 776343.

New Programmer's Toolkit

MMC AD have released the latest version of its Programmer's Toolbox (reviewed in the April '90 .EXE). New options, over and above the old lint, program flow analyser, pretty printer and cross-referencing utilities include an ANSI function prototype generation, advice on overlaying and virtual memory arrangements based on program structure, a nifty typedef to English converter, static and dynamic allocation information, and a general speed-up of all the main utilities. The C Programmer's Toolbox is available in the UK from System Science at £199.

Late Travel Update

A reminder that our TopSpeed-sponsored 'Win A Trip to CeBit Hanover' (see page 25 for details) closes on **March** *10, a scant two weeks away from the day this is published, and only three days before CeBit starts, so remember to include a phone or fax number where you can be quickly contacted.*

They Can Read Our Minds...

Sequiter software obviously knows a good idea when they see one: As .EXE went to press, it followed Bryan Boreham's example, and announced a C++ version of Codebase, their dBASE file management library. The package is called Codebase++ and is being distributed by The Software Construction Company (Tel: 0763 244114). It is priced at £190.

Demo II improved

Sage have upgraded Dan Bricklin's Demo II, the prototyping utility. New enhancements allow users to overlay text onto bitmapped images, the obligatory mouse support, VGA video support, autosave, and a SAA/CUA style guide. Demo II V3.0 is available in a single user pack for $249. A professional version includes the ability to encrypt files, and removes the initial Sage sign-on screen. This costs $995. Demo II can be ordered on 0101 301 230 3307.

ISYS in the UK

Isys is a award winning text retrieval system that has had a great deal of success in Australia. It works as a TSR, indexing every word in every document unless instructed otherwise. Users can then instantly search and retrieve a piece of information from the document database in a matter of seconds. No statistics are given for disc use, but the review copy showed a reasonable compaction rate. It is now being distributed in the UK by Equisys (071) 729 7447.

DOS compilers companies seeking new pastures

Announcements by the major DOS compiler vendors have flooded in this month. Borland, JPI, Microsoft, Watcom and Zortech have all announced (or pre-announced) major new directions. Hearteningly, each one has chosen to exploit a different area of programmer need.

Of these, the announcement of Borland C++ for Windows on will have perhaps the most impact, with its aggressive pricing taking the cost of a full Windows C development package, with compiler, developer environment, resource editor, assembler and debugger down to £299.95. Borland C++ is reviewed on page 33.

Microsoft has replied in the short term with a 25% cut in the price of the Microsoft C V6.0/Windows SDK pack, taking the official Windows route down to £549. What happens next is hard to determine: Microsoft's move was clearly a pre-emptive one, and the indications are that they did not expect Borland C++'s price to be quite *that* low. Nonetheless, Microsoft has some very big sleeves from which to pull new tricks - see 'Next Year's News'.

Meanwhile, other compiler manufacturers are taking different tacks. The optimising experts, Watcom, has announced that its Windows version, Watcom C V8.0 C/386 for Windows, will arrive in April. It's selling point will be its use of Windows' 32-bit interface for its flat-memory code model, and will offer the same features as Watcom's established DOS extender 32-bit compiler. Watcom can be quizzed on 0101 519 886 3700. Pricing will be around $2,250.

Zortech, in an update to last month's pre-announcement, has said that its SCO Unix C++ compiler is now shipping, as is its new 32-bit DOS compiler, based on Phar Lap's DOS extender technology. Zortech has stated in the past that its main intention is to provide source compatibility across as large a number of platforms as possible. Zortech can be contacted on 081 316 7777.

Meanwhile, JPI (071 253 4333) has been busy working on two new languages, and an update to their multiple-language compilation platform. The new JPI products will be announced at CEBIT in Hanover in mid-March. The languages are TopSpeed Pascal and C++. Each extra language will cost only £59 when added to the 'base' of the new TopSpeed Environment (also priced £59).

Windows Grabbing

Nothing is more slippery than a Microsoft standard file format, we say, and the .PCX graphics format is no exception. The version produced by Windows, for example, played all kinds of tricks on the software used to display .EXE's screen-shots. Enter the latest release of HotShot Graphics, a nifty little screen grabber which can work in Windows or DOS mode, and will happily convert from PCX, to TIFF, to GIF, to GEM, to MacPaint, to WordPerfect .WPG, to encapsulated PostScript. It can also print to a number of printers, including IBM, Epson and HP laser compatibles. HotShot Version 1.73 is what you should ask for: it's available from Ctrl-Alt Deli (0908 662759) for £179.95.

Mini Unix for the PC

Minix, like Coherent, is a cheap UNIX for those wishing to explore the operating system. The package costs £125+VAT, and is system call compatible with V7 UNIX. It includes a disk editor, vi emacs and ed clones, over 175 utilities including ar, make, dosread, nroff, kermit and zmodem, and a full K&R C compiler. Unlike Coherent, you also receive the fully commented source of the operating system and its utilities (excluding the compiler, which is available separately). The manual (which documents the entire system) contains instructions on how to re-MAKE your system from the source files. It can run on a 512 KB PC with a 10 MB disc drive. The new release even contains an Ethernet driver and Amoeba networking software, so you can create rlogin systems on a PC network. It's been around for four years now, and the new version, V1.5 has just been made available. Apart from its other unique features, Minix is also peculiar in that it is marketed as though it was a book. The publisher, Prentice Hall, recommends that you order it through your local book seller. Or you try your local library.

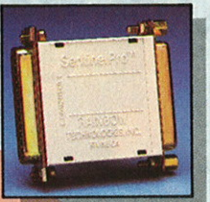
SCO Meet

The Santa Cruz Operation (SCO) is holding its first European Developer Conference on April 16 and 17 at the Frankfurt International Hotel. Presentations and hands-on workshops will be given by senior software engineers from SCO's own UNIX development team. The conference is being held as part of the company's Developer Programme, which programmers using SCO products can join free of charge. The conference fee has been set at £495. Details on 0923-816344.

With £1 Billion Worth Of Protected Software...

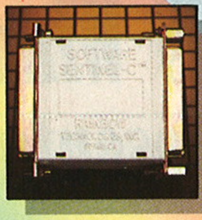
Rainbow Is The Safe Bet.

SentinelPro™



• Runs under DOS, OS/2 and Xenix • Algorithm technique (Never a fixed response) • External parallel port installation • Minimal implementation effort • Higher level language interfaces included • 100 times faster than fixed-response devices (1 ms) • ASIC design for reliability

Sentinel-C™

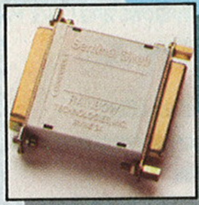


• Protects multiple packages with one device • 126 bytes of non-volatile memory programmed before shipment of the software • Rainbow supplies a unique adapter for programming the unit • Higher level language interfaces included • Runs under DOS, OS/2 and Xenix • External parallel port installation

Eve™

• For the Macintosh SE and II • Complies with Apple Desktop Bus Interface requirements • Rainbow-assigned developer passwords to prevent tampering by other developers or sophisticated "hackers" • 7 locks per key, usable individually or in combination, on one or up to seven applications

SentinelShell™



• Runs under DOS on IBM PCs and compatibles • Protects without requiring access to the source code • Completely transparent to the end user • User-friendly software • Pocket-size key attaches quickly to any standard PC parallel port • ASIC design for reliability

DataSentry™

• Completely user-installable • Pocket-sized external device • Menu-driven, user-friendly interface • Single- or multi-user security system • Audit trail, log-on identifiers and automatic encryption/decryption of entire directories • Secures data transmitted by modems • Prevents recovery of data by utility programs

Software developers creating the latest applications for the IBM PC/XT/AT, PS/2 and compatible systems can now turn to the Software Sentinel range of hardware keys for the *first-class* in *world-class* software protection.

There's the best-selling SentinelPro, known worldwide for its virtually unbreakable security, its ASIC technology and its invisible operation.

A close relation, the Sentinel-C for custom configurations, enables multiple package protection with a single device.

For the Apple market, security-minded Mac software developers can now secure their return-on-investment, too. Eve plugs into the Mac ADB connector and is completely transparent to the user–providing up to seven programmable security locks per key.

Rainbow's latest protection strategy is the SentinelShell–that lets users place a "shell" around existing, off-the-shelf programs. Access can be limited to those issued a key, so that libraries, educational establishments and corporations can very simply guard their investments.

Available soon from Rainbow is the DataSentry, a user-installed key that provides low cost security for sensitive data in both database applications and corporate/banking environments.

No matter where you sell your software worldwide, stay in control of your distribution and revenue by choosing the internationally accepted standard in protection... Rainbow Technologies. Be sure. Protect your pot of gold at the end of the rainbow.

CALL TODAY OR ORDER NOW.

☐ Please send me a SentinelPro Evaluation Kit.
I enclose £50 + VAT Payable to Rainbow Technologies Ltd.
☐ Please debit my credit card. Access ___ Visa ___ Amex ___
Expiry Date ________ (PLEASE TICK)
☐ Please send me more information.

Name ____________________
Position ____________________
Address ____________________
Telephone ____________________
Signature ____________________ EXE 3/91

RAINBOW TECHNOLOGIES LTD., Shirley Lodge, 470 London Road, Slough, Berkshire SL3 8QY.

RAINBOW TECHNOLOGIES

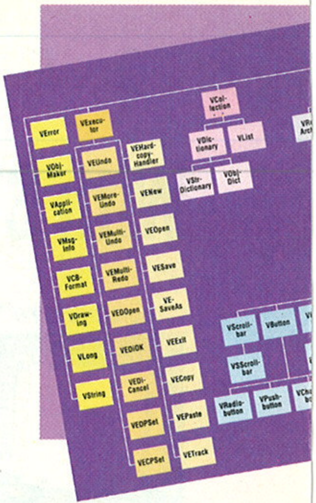
Shirley Lodge, 470 London Road
Slough, Berkshire SL3 8QY
Tel: 0753-41512 Fax: 0753-43610

CIRCLE NO. 552

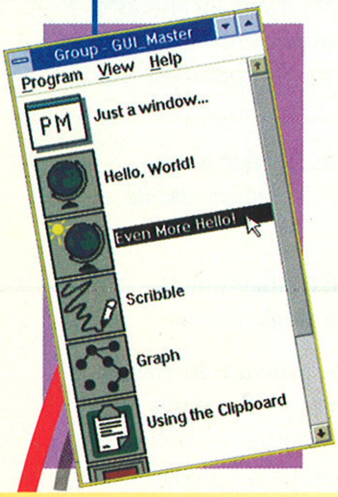
In the example programs you can see how *the GUI_MASTER* takes care of all the standard CUA functions like

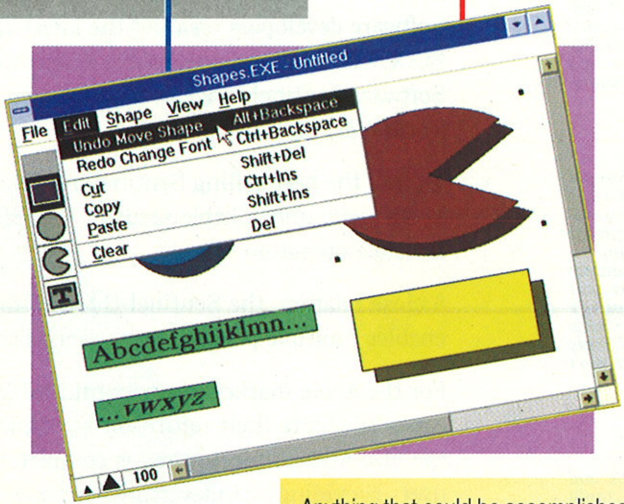
- Scrolling
- Runtime error handling
- Printing
- Undoing/Redoing the most recent user actions
- Text editing
- Standard mouse interactions such as dragging and sketching
- Archiving objects
- Standard dialogs such as open, save as, fonts, color picker
- Using the clipboard

The 7 example programs range from simple to fairly complex. Together with the cookbook we provide they help you gain insight in all the possibilities of the *GUI_MASTER (Class Tree for C++).* They come complete with fully annotated source code, so you can even use them as a basis for your own applications.



Effectivity MS Windows Presentation





If you are building sophisticated C++ applications why not benefit from our *GUI_MASTER (Class Tree for C++)* to give you

- a head start in building GUI applications
- a GUI toolkit consisting of 85 reusable classes, together forming a well-tested and fully functional 'standard application'
- platform independence between MS/Windows and OS/2 Presentation Manager

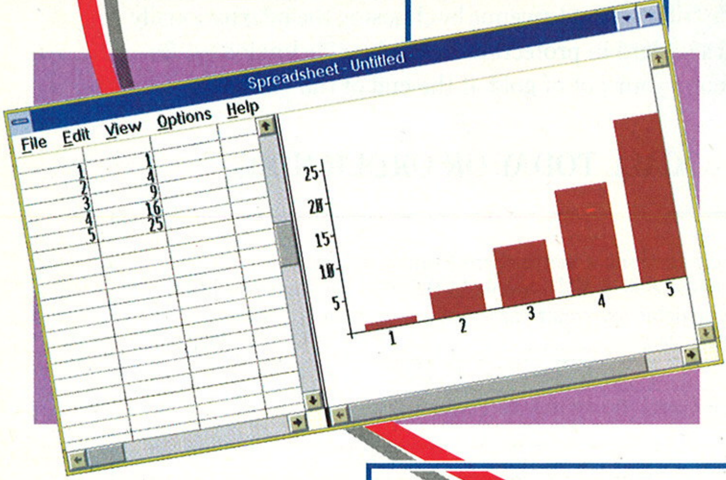
The product offers

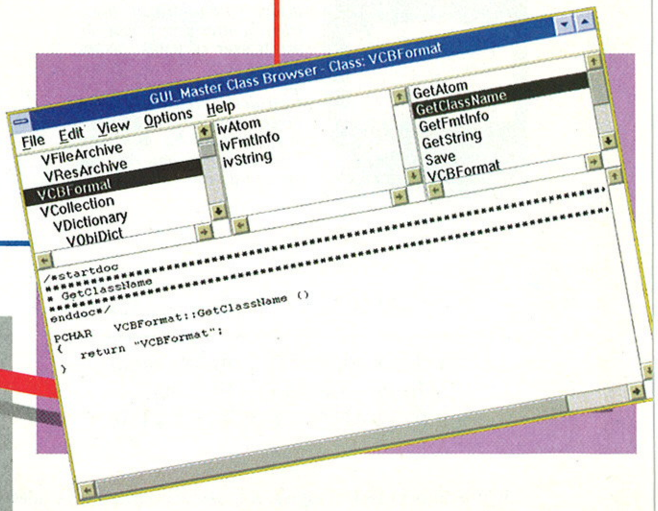
- 7 example programs
- a Class Tree
- a C++ Source Browser

The *GUI_MASTER (Class Tree for C++)* enables you to specify different worlds, different representations of the same data which can be shown in different parts of the same window. If you modify some of the data in one world this is automatically reflected in the other world.

If you use *GUI_MASTER* your application is, as a matter of course, structured around the Data/World concept. An excellent basis for GUI applications.

Anything that could be accomplished through 'normal' programming can be done if you use *GUI_MASTER*. It's just that *GUI_MASTER* makes it all a lot easier. Many of the things you might not include in your application because you think that they're too complex or time-consuming to program, are easy to make with *GUI_MASTER*.





The class browser is of crucial importance to enable you to view programs as collections of classes and not as a set of source files. It enables you to examine code made by others, and to understand the class structure of that code, thereby promoting the concept of 'the programmer as reader' and stimulating the reuse of software.

With our browser you can

- browse through the source code of any C++ program
- launch the editor of your choice to modify your C++ programs
- quickly locate any method in the Class Tree

Letters

We welcome short letters on any subject that is relevant to software development. Please write to The Editor, .EXE Magazine, 10 Barley Mow Passage, Chiswick, London W4 4PH. Unless your letter is marked 'Not for Publication', it will be considered for inclusion on this page.

Windows trouble

Sir,

You may be interested in our experiences trying to use a C++ compiler with Windows. As I'm sure you're aware, the only current viable alternatives are Zortech C++ and Glockenspiel with their AT&T C++ (V2.0c) port. I understand Borland's product will be available shortly.

The Glockenspiel product works OK, but it seems rather clumsy, being a preprocessor. Also (and this is important) it won't run in a DOS session under Windows Enhanced Mode, due to some problem with the Rational Systems DOS extender. This makes the development process very tedious, because you either have to start Windows afresh each time you want to make a test, or run Windows in Standard Mode (in which case swapping between a DOS session and Windows takes as long as starting Windows again!)

Zortech C++, on the other hand, is an integrated compiler and will run in an Enhanced Mode Windows DOS session.

There have been a number of revisions of V2.1 of the Zortech C++ compiler in the last few months. We upgraded from 2.0 to 2.13 some months ago. We then further upgraded to 2.17 two weeks ago and finally received 2.18 last week (fresh off the press).

My first tests under Windows with 2.13 were encouraging. I used the Zortech WorkBench (ZWB) in a Windows DOS session and generated a working program very quickly. However, I then noticed my floating point arithmetic didn't seem to be working and made some more detailed tests and checked with Zortech support. It turns out there are several reported problems regarding Zortech's floating point under Windows and their bulletin board carries various attempted fixes, none of which solves all the problems.

I received a call from an American visiting from Zortech US Tech Support, who acknowledged the problems and told me that Walter [Bright, Author of Zortech's compiler] was going to address them in a few months time, after he's completed the work on their multi-platform implementations.

Whereas under DOS, program development systems can be self-contained, with their own architecture and parameter-passing conventions, under Windows they have to follow what Microsoft provides. This may or may not be a whiff of monopoly here, but it's fact. Therefore, we at Bits Per Second have no choice but to go with Glockenspiel (however clumsy) until Microsoft's own C++ compiler is available.

By the way, I have no grumble with Zortech Tech Support - it has been consistently friendly and helpful.

John Marsh
Bits Per Second

Unfair to RISC

Sir,

I read with amazement the article 'I am not Spock...'by Jon Moseley *(.EXE, December '90)*. While the subject core had a very valid argument in that the various DSP devices do have an important place in the embedded market, for some reason the author decided to 'wrap' it with a heated attack on RISC technology. Why RISC was singled out for comparative discussion was very unclear.

At my previous employer, some years ago, I worked on systems known as 'bit slice' processor, for the most part in the area of DFTs. As the author knows, the PID was a main part of the required algorithms. Bit-slice was, in many ways, the forerunner of RISC: simple instructions that executed in a single cycle. The speed of RISC is not as 'new' as it is often portrayed. It takes its performance, just as bit-slice did, not from the 'clever tweaks' the article refers to, but by carefully matching hardware and software.

Simply put, RISC processors can be predictable, even more so than many CISC or DSP processors. On our own processor, the Am29000, interrupt response is predictable (and quick), and a freeze mode ensures that latency is reduced and bandwidth maintained for all interrupts and traps. John Moseley's assumption that RISC is unpredictable is based, I think, on the incorrect assumption that RISC and cache are indivisible. The Am29000 does not require a cache to achieve single cycle instruction length. Certainly, it is available, but this is only to address the latency of instruction fetching, *not* the bandwidth.

I felt it was a shame that a very informative discussion on the use of DSP had to be wrapped in such emotional wording.

Dave Everitt,
Advanced Micro Devices

Exception to Rule

Sir,

Is P G Rule serious? *(Letters, Feb 1991)*. If he is really a Senior among Software Engineering Consultants then he should be offering us answers to his questions or, at least, riding some favourite hobby horse of his own.

The key to Rule's major difficulty lies in his assertion 'From the problem description I should then be in a position to derive a solution'. This assertion may be true for him, but it cannot be true for that ideal specification language which he is seeking.

First we have to describe the problem. Even a Mickey Mouse problem requires great effort to describe it in full. Large, real problems' descriptions require disproportionately greater effort because of the increased number and types of interactions among the parts. If we are to automate the generation of programs, we surely have first to write their requirements in a formal language. While there are such languages around (eg VDM, Larch, LOTOS and Z) and they severally fit a wide range of computing problems, I haven't yet seen how they allow the painting of a screen layout and I haven't heard of their successful use in generating solutions to real world problems. (And don't ask how the user validates them unless the language processor can also generate a plain language commentary.)

Then we have to solve the problem. Being in a position to derive a solution depends on who the 'I' is. Do I know the problem domain? Have I seen apt solutions for similar problems? Do I know a man who has? It is a belief of the otherwise uninformed reader of science fiction that the process can be done by an automation.

Finally, where do we write the myriad details which are part of both the problem and the solution - its constraints and sizings, the links to existing systems? Think carefully before you answer.

Rule has a minor point about programming languages. I am not at all disappointed that the relative merits of programming languages are still being discussed. Different languages suit themselves to different classes of problem - that's why they were developed - and they provide different levels of comfort and safety to programmers.

Now suppose we did have a method of deriving a solution directly from a specification; the solution would of course be stated in some formal language so that it could then be proved and be translated into executable code. At this stage the choice of language is relatively unimportant. The results of generating a program from an arbitrary specification or of parameterising a fixed-logic 4GL is a lump of executable code. The intermediate language does not have to be comfortable or safe since no person is going to look at it; it could be machine code.

The supposed library of reusable parts is a useful notion; each part would contain both the definition of some small problem and its solution - it's a knowledge base (use Prolog?). Because all problems are different, and not only in detail, it would have to be huge like the vocabulary and grammar of a spoken language; its 'brain' would have to be supremely capable like a combination of Wernicke's and Broca's areas in the left of ours.

In common with Rule, I use a simple battery of practical, essentially paper-based tools when developing software. I don't have an answer to his basic question but I think I have a more realistic view on the question itself.

Richard H Pickard
Bedford

There were the bugs

Sir,

How kind of you to publish our C compiler's bugs! *(February 1991)*. Happily for us, however, these bugs are not in our latest release, nor in our Version 3, due out any day now.

Careful readers will have noticed a couple of mistakes in the test program from Doru and Dan; those that did will have corrected them already so I won't waste space with a listing.

With regard to the ANSI standard, we are the proud owners of that rare document, the validation certificate for ANSI X3.159-1989 (the C Programming Language). Surely this alone wins us a .EXE T-Shirt!

Seriously, though, thanks for the evaluation and kind words about our compiler.

Vincent Jones
Senior Analyst
Quality Assurance and Technical Support
Jensen & Partners International

The errors that Vincent mentions are a result of sloppy editing by WRW; apologies for this.

EXE

Letters submitted to this page may be edited. The writer of the best letter of the month, as judged by the Editor, will be rewarded by a T-shirt or similar-valued .EXE trinket. The best letter is the one printed first.

Adding a C++ front-end

CodeBase 4 is a commercial library which provides dBASE data access to C programmers. However, Bryan Boreham is a C++ programmer.

I'm often asked, 'This object-oriented programming seems all very well, but what can it do for me?'. There are many standard answers: code reuse, modularisation, extensibility etc, but they are somewhat glib, and most companies can't afford to re-implement whole systems using the new paradigm, let alone spend the time to learn what it is all about. OOP requires a whole new mind-set, and is a little too young to be furnished with hundreds of useful 3rd-party libraries in the way that C and Turbo Pascal are.

C++ is the object-oriented language of the moment, strongly hyped on all sides by compiler and library vendors. It is certainly the most widely available OOPL and the easiest one for most people to start using, since it will link with their existing code and the syntax is already familiar to C programmers. But there are problems with C++ as the route to OOP. Today's C++ compilers are based on 1960's technology, going through an often lengthy compile and link cycle between program runs, and generally falling far short of the powerful integrated environments provided by Smalltalk and its clones.

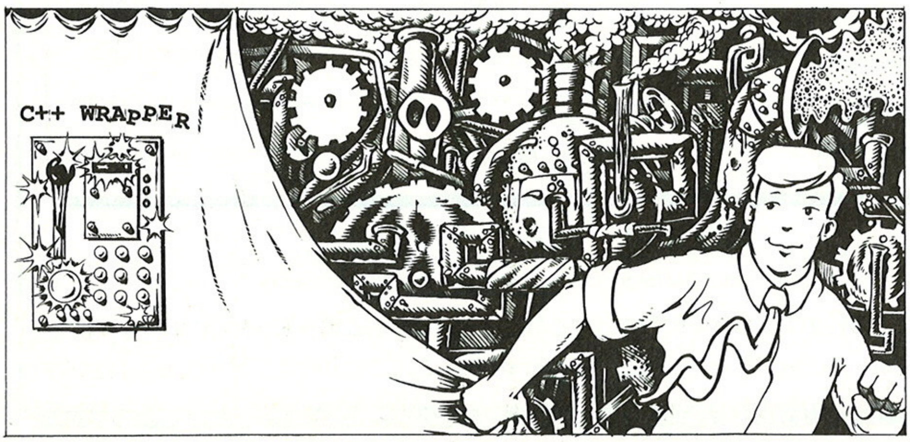
However, C++ isn't only about OOP. It truly is 'a better C', providing strong type-checking and extending the syntax in many small but useful ways, as well as very powerful data abstraction. In this article, I hope to demonstrate how C++ can be used to improve the working environment of C programmers by small steps, without the need to throw away all your existing ideas and code.

CodeBase

Recently, I was building an interactive workshop-scheduler in C++, using CodeBase 4 from Sequiter Software to access my Clipper data files. CodeBase is a C library, and so the more I worked with it, the more I remembered what I hate about working with C.

I don't want to print the whole scheduler program to illustrate my point, so instead I have adapted a short example from the file D4SIMPLE.C which comes with CodeBase (see Figure 3).

First, a quick description of dBASE files, in case you've never worked with them: each record is a fixed size, and made up of a fixed number of fields. In the example, each record contains three fields, each of which holds an integer number (see Figure 1). The record structure, ie the name, type and size of each field, is stored at the beginning of the file, in the header (Figure 2), which means that files can be adapted or extended as a system matures without necessarily requiring recompliation of each program. However, it does also require that programs which use dBASE files always start by interrogating the file header to sort out where the data will be, and this can get messy without proper language support. The Code-



A_VALUE	B_VALUE	SUM
15	23	38

Figure 1 - The structure of each record in the example

Base library hands out reference numbers which are used to refer to files and fields which the C programmer must keep track of.

The example merely reads each record, adds the two values A and B together, and stores the result in field SUM. CodeBase is very helpful in decoding the file header and converting values, but the code required is quite complex for this simple task.

If you read the program in Figure 3, I'm sure you will see several problems. In order to avoid name clashes with other libraries or with the user's code, the library writers have prefaced every routine with 'd4' or 'f4', but this obfuscates the meaning of each call. The operation of the program is clouded by the numerous function calls, and it is easy to get the arguments in the wrong order, say for the `f4replace()` call. With a more complex example, using more than one database file, things can get quite hairy.

The example works directly with the returned field values, but in order to do serious work with the data, it is usually necessary to hold onto the field reference number in one variable, then pull the actual value out into another before starting work on it, which complicates matters even further. As I got bored writing yet another `f4ref()` call one day, it suddenly hit me that C++ could work wonders on the complexity, and hide all those nasty C functions beneath a simple interface layer.

The C++ version of the above example is presented in Figure 4, and I hope the fact

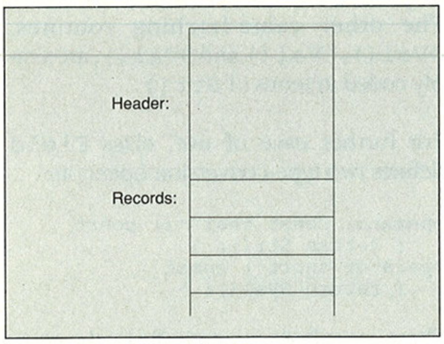


Figure 2 - How dBASE files are organised

that it is half as long will interest you enough to read on and see how it's done.

C++ Interface

I defined a class `Field` to hold the reference number and do the dirty work of extracting and assigning values. The constructor does the work of finding reference numbers, and various operators are defined to make `Field` objects function much like normal data variables. The full class definition is presented in Figure 5.

I'll make a couple of technical C++ points about the design of the class. First, the data members really are private: there are no public 'access routines' to get at them. The facilities provided by the class should be sufficient for all uses, and no-one should have direct access to the reference number or actual location of the data, although it is possible that a `Length()` method might be useful. Second, anywhere that I don't intend to modify data, I write the word `const`. This might seem slightly overdone, the `const`s almost outnumbering the identifiers, but I feel that if the language can protect a programmer from himself, it should be allowed to. *(Editor's note: Bryan frequently uses* `const` *after a member function and before the body. For the benefit of anybody else whose C++ syntax is still a little hazy - frankly, I had to look this up - it is a nifty Release 2 feature which declares that the function does not modify the object's instance data.)*

Along with the `Field` class comes class `DB` (see Figure 6), encapsulating all access to the underlying dBASE file. All of the methods for class `DB` are quite straightforward, merely passing arguments on to CodeBase calls, sometimes checking results or remembering some state information, so I won't go into detail here (the source code is available on request from the .EXE offices). Instead, I'll concentrate on the more interesting `Field` methods.

Initialisation

The public constructor for class `Field` (see Figure 7) sets up all the data members by calling CodeBase routines, then checks the results for validity. If you're just starting to use C++, you might not realise that ordinary member variables like `int` and `char*` can be initialised in this way, but in fact this is the most effective way to do it, since it allows them to be declared `const`. In effect, this says to the compiler: 'don't let me change the value once it's been set up'. One subtlety is that member variables are initialised in the order of declaration, irrespective of the order in which initialisers are written in the constructor, and so `ref` must be declared before `ptr` and `len`, since it is used in their initialisations.

This implementation is arranged quite neatly so that no code is required to say which database each field belongs to; simply declare all the `Field` variables right after the appropriate `DB` instance and the `DB` constructor will set things up so they get correctly attached.

```
long    a_value, b_value, sum ;
int     rc ;
double  result ;

/* Open the file and read the header. */
d4use( "SIMPLE" ) ;

/* Obtain numbers to refer to each field by. */
a_value =  f4ref( "A_VALUE" ) ;
b_value =  f4ref( "B_VALUE" ) ;
sum     =  f4ref( "SUM" ) ;

/* Start at the top of the database, and loop until we get to the end */

for ( rc = d4top(); rc != 3; rc = d4skip(1L) )
{
   /* Get numeric values from the record, using the reference numbers */
   result =  f4value(a_value) + f4value(b_value) ;

   /* Stuff the number into the other field */
   f4replace( sum, &result ) ;

   /* Write back the record */
   d4write( d4recno() ) ;

   /* Move on to the next record */
   rc = d4skip(1L);
}
```

Figure 3 - The example program in C

Evaluation

The `ptr` member of a `Field` points into the database record buffer, where the data for all fields in that database is stored in the standard dBASE format, ie straightforward ASCII, padded to the full length with spaces. This presents something of a mismatch with a C-based language, where strings are null-terminated and the trailing spaces are probably unwanted. To get round this problem, class `Field` declares a method `Str()`, which returns the field's value as a C string.

It would be nice to be able to implement this by simply finding the first trailing space and storing a zero into it:

```
const char *Field::Str() const
{
  char *p = ptr + len - 1;
  while (*p == ' ')
     *p-- = 0;
  return p;
}
```

Unfortunately, this falls down when the field fills the space exactly and there is no trailing space, and it also alters the record buffer in a way that may cause trouble if that record is written back to disk with the null byte still in place. So, `Str()` must make a copy of the data and return that.

If I allocate bytes for the copy using `new`, then they must be deleted somehow, but this is awkward for users of our `Field` class, and impossible to arrange automatically, so the solution is to provide a static buffer, into which the string is copied. In fact, CodeBase itself has such an arrangement, but there is a problem. Using such a buffer places restrictions on the way in which the results of `Str()` can be used, because subsequent calls might overwrite the contents, and it is uncertain what will result from something like:

```
if (strcmp(f1.Str(),f2.Str()))
  break;
```

Really, this indicates a basic deficiency in the C language, which C++ inherits directly. The exclusive use of a class `String` giving automatic storage management would solve this problem, but that is quite a dramatic move. Every existing C library routine that deals with `char*` strings would have to be replaced or encapsulated with one that dealt with `Strings`, and since this article is about using C++ incrementally to help make use of existing code, a different approach is required.

The solution I adopted (see Figure 7) was to use a bigger buffer which can hold a number of field copies (say 20), and use them in round-robin fashion, on the assumption that no-one will want to use so many `Str()` results in the same expression.

```
DB simple ("SIMPLE");
Field a_value("A_VALUE"), b_value("B_VALUE"), sum("SUM");

for ( simple.Top(); simple.Status() != 3; simple.Next() )
{
    sum = a_value.LVal() + b_value.LVal();
    simple.Write();
}
```

Figure 4 - C++ version of example program

```
class Field
{
private:
    const long ref;         // Reference number from CodeBase
    char * const ptr;       // Pointer into record buffer
    const int len;          // Length of field

    Field (long, char * const, int);

public:
    Field (char *name);

    /* Explicitly convert to various types */
    const char *    Str()           const;
    double          DVal()          const;
    long            LVal()          const;
    short           SVal()          const;

    /* Implicit type-conversions */
    operator const char *() const { return Str(); }
    operator short()        const { return SVal(); }

    /* Assignment from various types */
    Field& operator=(const char*);
    Field& operator=(const double);
    Field& operator=(const long);
    Field& operator=(const short);

    /* Comparison between Fields */
    int    operator==(const Field &f) const;

    /* Indexing into an array */
    Field  operator[](const int) const;
};
```

Figure 5 - Full definition of class Field

```
class DB
{
    int ref;                        // CodeBase reference number
    int iref;                       // Reference number for index
    int status;                     // Return-code from last operation

public:
    /* Constructor takes database name and optional index name */
    DB (const char *name, const char *iname = NULL);

    ~DB();                          // Destructor closes file

    int Status() const              // Return status of last call
      { return status; }

    void Select();                  // Select this database
    void operator()()               // Shorthand for Select()
      { Select(); }

    /* Various ways to move to another record */
    void Top();
    int  Skip(long num = 1);        // Skip 'num' records
    int  Next();                    // Move to next non-deleted record
    void Goto(long recno);
    int  Seek(const char *key);

    /* Write data, defaulting to current record */
    void Write(long recno = -1);

    int  Deleted() const;           // Is the current record deleted?
    long Count() const;             // How many records in database?
    long RecNo() const;             // Return the current record number

    /* Clear out several records from the database */
    void Zap(long from = 0, long to = LONG_MAX);
};
```

Figure 6 - Class DB

The other value-fetching routines, `DVal()`, `LVal()` and `SVal()`, are simply coded in terms of `Str()`.

For further ease of use, class `Field` defines two type-conversion operators:

```
operator const char *() const
   { return Str(); }
operator short() const
   { return SVal(); }
```

These are called automatically by the compiler whenever a `Field` variable is used in a context where one of the target types is valid and type `Field` is not. Suppose we want to call `strcmp()` with the string values of two `Fields`, as in an earlier example. The compiler knows that the arguments of `strcmp()` must be of type `const char *`, and `Field` provides exactly that conversion, so the following code is much more clear and succinct, yet does exactly the same job:

```
if (strcmp (f1, f2))
  break;
```

I didn't go overboard on implicit type-conversions because they can become ambiguous - if conversion operators for `short` and `long` integers are both defined, how should the compiler decide which to call when a `Field` appears in an arithmetic expression?

Assignment

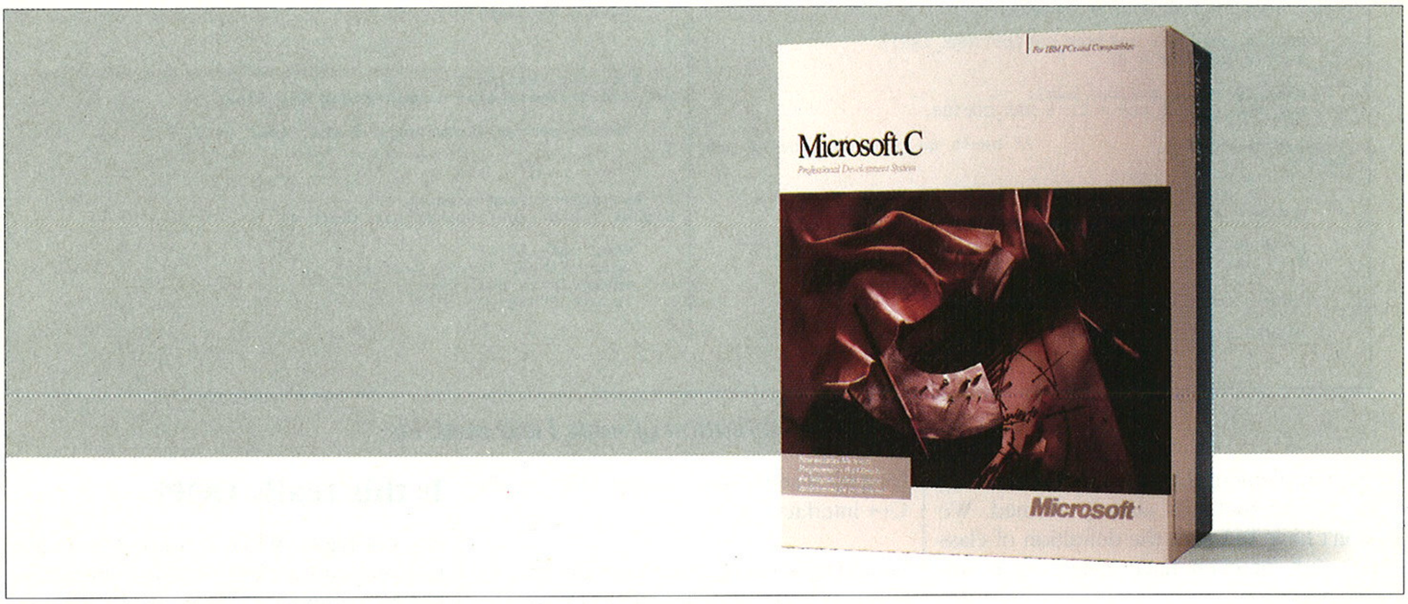
A full set of assignment operators is provided so that `Fields` can be set from `char*`, `double`, `long` and `short` values. This conversion is never ambiguous; the compiler always calls the one closest to the type presented.

Class `Field` doesn't currently check to see what type the database field was declared as. I figure that it is useful to be able to assign a number to a string field without undue contortion, and it fits with the general lightweight style of this interface. It is not too hard to implement since CodeBase provides a routine `f4type()`.

Support for more types can be added by adding new assignment and type-conversion operators to class `Field`, or by declaring them as friend functions in other classes. For instance, if we have a `Time` class, storing hours and minutes, we might add to its declaration two new friend functions:

```
friend Field& operator=
    (Time &t, const Field &f);

friend Field& operator=
    (Field& f, const Time &t);
```

C Breeze.

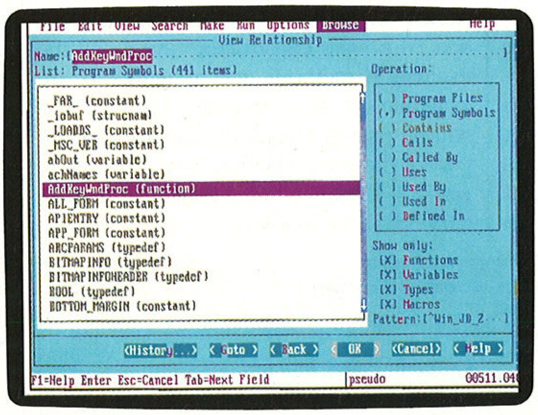


Get set for smooth sailing. Because Microsoft® C Professional Development System version 6.0 is not only designed to be fast and powerful, but also easy to control.

Thanks in large part to Microsoft's unique Programmer's WorkBench.

It integrates the debugger, editor and compiler, allowing them and all the other powerful programming tools to share data.

Tools such as our Source Browser, that taps directly into the project database. This feature will give you instant access to the relationships between functions, macros, types and variables. It'll even give you a full call tree that literally draws you a map.



Explore the relationships between functions, variables, types, and macros using information created by the compiler and stored in the project database.

With our CodeView® Debugger, you'll save a lot of time. This powerful tool not only debugs any size DOS* or OS/2 application program, on any 286 or 386™ machine, but also lets you explore nested structures and arrays.

The programmer's WorkBench even has an open architecture that is supported by many third party tools.

Our online and hardcopy documentation are designed to complement each other. Together they provide you with an intelligent mix of professional guidance.

You'll find the C Advisor always online and ready to help. It is a hypertext-based reference system complete with sample coding solutions you can copy and paste directly into your program.

You will also find the hardcopy documentation offers many useful tips and techniques for advanced C programming.

So whether you're developing applications for the MS-DOS® Microsoft Windows™ version 3.0 or MS® OS/2 Presentation Manager systems, our Programmer's WorkBench will allow you to write the fastest code around in the shortest amount of time possible.

For a free white paper with more details, just call Microsoft on (0734) 500741.

Then get your hands on Microsoft C version 6.0. It'll blow you away.

Microsoft®

Microsoft Languages and Development Kits may be purchased from your usual dealer or direct from the following Microsoft Language Resellers: Grey Matter, 0364 53499. Software Construction Company, 0763 244114. System Science, 071 833 1022. QA Training, 0285 655888. *This requires 550K of available extended memory.

CIRCLE NO. 555

```
/*  Public constructor for class Field  */
Field::Field (char *name) : ref(f4ref(name)), ptr(f4ptr(ref)),
                                    len(f4width(ref))
{
    if (ref == -1)
        Fatal ("Field %s not found", name);
    if (ptr == NULL)
        Fatal ("Field %s ptr NULL", name);
}

/* Conversion to a zero-terminated char* string */
const char *Field::Str () const
{
    static char str_buf[MAX_COPIES][MAX_LEN];
    static buf_num = 0;

    char *p = str_buf[buf_num];
    buf_num = (buf_num + 1) % MAX_COPIES;

     int ln = len;              /* Don't go over the end of the
buffer */
    if (ln >= MAX_LEN)
        ln = MAX_LEN - 1;
    memcpy (p, ptr, ln);

    /* Work backwards to find the last space */
    for (char *q = p + ln - 1; *q == ' '; q--)
        ;
    *(q+1) = 0;                /* Null-terminate */

    return p;
}

/* Conversion to a double */
double Field::DVal() const
{
    return atof(Str());
}

/* Conversion to a long */
long Field::LVal() const
{
    return atol(Str());
}

/* Equality testing */
int Field::operator== (const Field &f) const
{
    return memcmp (this->ptr, f.ptr, len) == 0;
}

/* Assignment from a short */
Field& Field::operator=(const short x)
{
    char buf[128];
    sprintf(buf, "%*d", len, x);
    memcpy(ptr, buf, len);
    return *this;
}
```

Figure 7 - Implementation of some Field methods

then implement this in terms of the string or integer methods already defined. We don't have to touch the definition of class `Field`, or even have access to its implementation.

All the C comparison operators could be defined on class `Field`, and implemented in terms of `memcmp()`. I have only illustrated how this is done for the equality operator, in the interests of simplicity.

Arrays

My scheduler application had one file where each record contained 52 fields giving hours available for each week in the year (see Figure 8). I reckoned that this sort of arrangement, where the same type and size of field is repeated a fixed number of times within a record, occurs often enough for it to deserve special support within the C++ interface layer.

I could have written variants of the `Str()` and `LVal()` methods which took an extra index parameter, or given the standard ones an index which defaulted to 0, but that would complicate those methods, and slow down processing in the majority of cases. However, C++ lets me overload the array-index operator (Figure 9) in such a way that any `Field` can be treated as an array. This method constructs a temporary `Field` instance on the stack of the caller using a special private constructor, and so all the evaluation methods and assignments are available on this temporary `Field`, and will be passed through to the appropriate part of the record buffer. Now I can write `week[32]` anywhere I could use an ordinary `Field` variable.

CAL	WEEK 1	WEEK 2	WEEK 3	WEEK 4	. . .
A	40	40	37	40	. . .
B	80	80	80	60	. . .

Figure 8 - Repeated week-data within a single record

```
Field::Field (long r, char * const p, int l)
        : ref(r), ptr(p), len(l)
{
}

Field Field::operator[](const int i) const
{
    return Field (ref, ptr + len * i, len);
}
```

Figure 9 - Field methods to deal with arrays

Is this really OOP?

No, not really. While it is true that in any C++ program using this interface layer there will be a number of objects of class `DB` and `Field`, they make no use of inheritance or dynamic binding, and hence should be described merely as Abstract Data Types.

To try to retain my object-oriented credentials, I should mention that I did find a way in which inheritance worked extremely well with these classes: after working with the basic `DB` implementation for a few weeks I wrote a subclass `SuperDB` that would read many records at once from the file, thereby speeding up networked access considerably for non-indexed, read-only databases. Because the interface was designed with strict encapsulation, this improvement could be written and tested in isolation, then slotted in with the minimum of alteration to code that already worked with class `DB`.

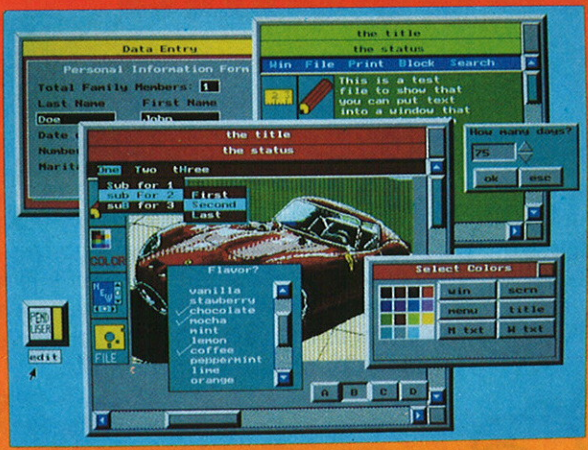
So, what do we have? A few hundred lines of C++ that make life a lot easier when using CodeBase, and, I hope, some techniques that you may find useful when writing your own tools in C++. It's not an object-oriented database by any means, but that's not the point. C++ is the way of the future, principally because it allows an easy transition from existing code and designs, and that is exactly what these classes provide.

EXE

Bryan Boreham is a software developer and consultant specialising in OOP and GUIs. He can be contacted on CIX as 'bjb'.

NEW
Genus GX Support

graphics-MENU 3.0



A sophisticated graphical user interface for Turbo Pascal, Turbo C, Turbo C++ & Microsoft-C.

Automatic use of EMS (expanded memory) if it is available to save the underlying image.

MENUS: several types including horizontal & vertical nested to any depth, supporting keyboard and mouse. Auto save of underlying image.

ICONS: beveled or not, can be used as buttons or as application holders. Can be automatically aligned within a window or isolated.

BUTTONS: automatic arrays allow easy placement in or out of a window. Multiple modes allow use as radio buttons or check boxes.

WINDOWS: optional beveled look. Automatic drag and resizing. Vertical and/or horizontal scroll bars. Title and status lines - easily changed.

FORMS: true data-entry forms available with add-on module. Data validation, range-checking, pop-up menus, help lines, dBASE-like picture strings.

CASE TOOLS: on-screen design tools for complete code generation of menu structures and data-entry forms.

SOURCE: library source is available in all languages.

TASKING: Fore & background tasking capability allows GUI tasks to proceed as a background activity.

Multiple Graphics Environments: Works with BGI (Borland Graphics Interface), Microsoft native graphics, Genus GX graphics or MetaWindow (from Metagraphics Software Inc)

NO RISK: 30-day money back warranty, free technical support, no licensing, no royalties.

£179 graphics-MENU
£379 graphics-MENU with source
£ 99 Data-Entry
£224 Data-Entry with source
£549 ProPak (full source & lib's)
Specify variant: BGI, Microsoft Graphics, Genus GX Graphics or META
Languages: Turbo Pascal 5.5 and 6.0, TurboC 2.0, TurboC++, Microsoft 5.1 (META only), and MicrosoftC 6.0 (META or MS graphics).

Island Systems

Highland Grafix Ltd
Telephone: 0294 69050
Fax: 0294 68286

DEMO AVAILABLE

VISA and MASTERCARD accepted

CIRCLE NO. 556

Compiler Performance

- Based pointers access far data with size and speed of a 16-bit pointer.
- Register-based parameter passing dramatically improves performance of function calls.
- Optimization pragma lets you control what optimizations are in effect from within your code.
- Integrated inline assembler.
- Dramatically improved local code generation.

Programmer's WorkBench

- New Programmer's WorkBench 1.1 is faster than ever before.
- Complete integration of edit, make, debug cycle.
- Source Browser gives you access to information on all aspects of your source code.
- Microsoft C Advisor provides fingertip access to the environment, C Language, and C Runtime Libraries.
- Under OS/2, Programmer's WorkBench takes advantage of OS/2 virtual memory protection, multitasking, and multiple threads to provide you the most productive development environment available.

CodeView version 3.1

- New version 3.1 is compatible with VCPI managers – CodeView can take advantage of extended memory along with VCPI applications like 386-Max.™
- Completely redesigned user interface providing a multi-window, multi-file environment with views into source, data, local and memory.
- Debugs nearly any size application on 286 or 386 machines.
- CodeView takes only 15K from the DOS 640K address space.

Documentation

- Advanced Programming Techniques: Tips and techniques for professionals – to help you get more out of C 6.0.
- C 6.0 Reference: Reference guide covering options and reference to runtime library routines.
- Advisor: Complete reference material online at your fingertips complements the C 6.0 Reference.
- Installing and Using: Gets you up and running with the Professional Development System.

Microsoft®

CIRCLE NO. 557

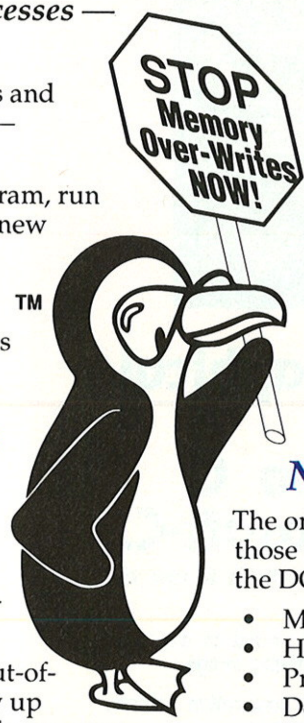
POWER DEBUGGING

BOUNDS-CHECKER

Finds out-of-bounds memory accesses — AUTOMATICALLY.

Flush out those Nasty pointer problems and other out-of-bounds memory accesses — AUTOMATICALLY.

Each time you make a change to a program, run BOUNDS-CHECKER while testing the new code. If you accidentally access out-of-bounds memory, BOUNDS-CHECKER will pop up displaying the offending SOURCE LINE. And your program runs at full speed.



> *"BOUNDS-CHECKER and Soft-ICE make sophisticated use of the most powerful versions of Intel's processor family to track down some of DOS programming's most insidious bugs. If you're developing programs for DOS, these are essential tools."*
>
> PC Magazine
> *July, 1990 pg. 48*

Ship Bug-Free Products

You can run BOUNDS-CHECKER while testing your program. There are no additional steps to your testing cycle, but you can feel secure when the program has passed through BOUNDS-CHECKER with no reported problems.

Many over-write problems and other out-of-bounds memory accesses do NOT show up during normal testing. An out-of-bounds memory location may be modified, but that particular location doesn't happen to be important at the time. Once the program is in the field and a certain network is loaded or a certain T&SR or device driver is loaded, that memory location suddenly becomes very important... AND THE SYSTEM CRASHES.

You can prevent these problems by making BOUNDS-CHECKER a standard part of your testing procedure.

Gives you the protection of a protected operating system under MS-DOS.

BOUNDS-CHECKER uses the 386 virtual machine technology to provide real-time memory protection. In addition BOUNDS-CHECKER uses the symbolic information output by your compiler to differentiate CODE and DATA. When your program is running, BOUNDS-CHECKER protects the program's CODE and all memory outside your program.

Soft-ICE 2.5
New Version, New Features

The only debugger specifically designed to solve those problems unique to MS-DOS that we call the DOS Nasties.

- Memory over-writes
- Hung programs
- Program too big to debug
- Debugging T&SRs and Loadable Drivers
- Multiple Symbol Tables
- Supports Microsoft C 6.0 & Turbo C++

Windows Developers

Run CODEVIEW for Windows on a Single Monitor

Nu-Mega has done it again!

CV/1™ $129

Call Today, Have it Tomorrow

BOUNDS-CHECKER	*$249*
Soft-ICE 2.5	*$386*
MagicCV 3.0	*$199*
CV/1™	*$129*

Special Offer...

Buy BC & S-ICE	***Save $100***
Buy S-ICE & MCV	***Save $86***
Buy all three	***Save $186***

Requires 386 PC.
MS-DOS is a trademark of Microsoft Corporation.
386 is a registered trademark of Intel Corp.

Nu-Mega TECHNOLOGIES INC

International Shipping charge is $40.00
Credit Cards or Wire Transfers only
30 Day Money-Back Guarantee

CALL TODAY 0101 603 888-2386 or FAX 0101 603 888-2465
P.O. BOX 7780 ■ NASHUA, NH ■ 03060-7780 ■ U.S.A.

CIRCLE NO. 558

Check Out the Library

Many programmers avoid using commercial code libraries, unwilling to pay for something they could write. Peter Anderton makes a case for bought-in libraries and explains the role of the non-profit making Numerical Algorithms Group.

'We can't afford luxuries like commercial software libraries. They're fine for mathematical boffins drawing plump R&D grants who want to do multidimensional non-linear optimisation thingy - but of no use for the likes of us. Last time I used one was back at college, when you had to get the key to the FORTRAN 66 punched card cupboard from the night porter...'

Do you associate technical software libraries with white-coated laboratory technicians and the 3B (Big, Bad and Batch) era of computing? Then you should know that things have moved on. There is now available a good selection of software in a wide range of non-FORTRAN 66 languages on many diverse platforms. And while they are still indispensable for your heavy number crunching and teaching applications, there is much that is of use to the general programmer.

Why not?

There are three common reasons why people don't use commercial libraries. The most obvious is that they simply didn't know that a routine already existed, in the right language and on the required hardware, to perform the operation in question. The second is the 'not invented here syndrome'. Everyone is guilty of this to some extent - we are about as likely to admit that someone else can program better than us as confess deficiency in either of the things people do in cars. However, have you ever moved into a house that has been owned by a Do-It-Yourselfer who couldn't? Third: the library will cost - not just time but hard-earned cash. If the convenience and time-saving alone does not justify the purchase price, why buy software in?

Well, because library software provides an off-the-shelf solution, it removes any uncertainty about completion dates. A good library is also reliable. The cost savings that can be achieved by using tried and tested software - letting someone else worry about achieving high speed, reliability and accuracy - are a less obvious benefit. If you can cut the time you have to spend developing and testing the building blocks of your program, you can concentrate on the high-level 'value added' parts of the system - which is where your competitive advantage lies.

> ***NAG's 'not for profit' charter means that all surplus cash is re-invested in the software***

Introducing NAG

After all that harping-on about the advantages of libraries, it is time to declare my interest. I work for NAG - the Numerical Algorithms Group - which was formed in 1970 as a joint project of the Universities of Birmingham, Leeds, Manchester, Nottingham and Oxford together with the Atlas Computer Laboratory (now part of the Rutherford Appleton Laboratory). These institutions pooled their resources to produce subroutine libraries in FORTRAN and ALGOL for the (then new) ICL 1906 computer. The Group was quickly a success, and soon other Universities and research bodies took on the libraries. From the start, NAG Ltd's 'not for profit' charter has meant that all surplus cash is re-invested in the software. There are no shareholders and, therefore, no dividends to be paid.

Historically, NAG's main customers were the Universities and Colleges, and it is here that most people first come into contact with NAG products. Yet more than half of NAG's current clients are in the private industrial and government sectors, and the demand for libraries in new languages and application areas has encouraged an ever-increasing range of products. The company has now become an international organisation, with subsidiaries in the USA and Germany. In Oxford itself, NAG Ltd currently employs over 100 people, including several world experts in fields such as linear algebra, numerical analysis and computer languages.

We are probably best-known for our FORTRAN Library. This now contains nearly 1000 routines and is currently in its 14th version, with chapters on subject areas including linear algebra, optimisation and ordinary differential equations. Apart from FORTRAN, our products which can be grouped into three categories: general-purpose libraries, specialist libraries and software packages. Among the languages that we support are Ada, C, FORTRAN, Pascal and Borland's Turbo Pascal. Each general-purpose library has a common mathematical and statistical theme, but with a different flavour reflecting the shortcomings of the particular language; for example, matrix operations, I/O routines, random number generation and standard mathematical functions, together with routines for returning machine constants and date/time utilities. The idea is that the programmer can build systems using an established portability base. All the code is designed

and written 'from the ground up' in the target language to maximise performance and ease of use.

There are also libraries specifically designed for smaller machines, and header files to allow C programmers to use the FORTRAN routines. Specialist products includes libraries for control engineering, linear programming, finite elements, data approximation and sparse matrices, and the portable graphics libraries include general graph plotting, curve fitting, contouring surface views and data presentation. We also supply mathematical and statistical software packages for users who want results without programming.

NAG Libraries

NAG FORTRAN Library. Now in its 14th edition and contains 889 numerical and statistical algorithms. Widely used in education, industry and government.

NAG C Library. A suite of routines covering a range of mathematical and statistical functions. Available on PCs, workstations and mainframes.

NAG C Header Files.A collection of header files for accessing the FORTRAN Library from C.

NAG Ada Library. Reusable Ada packages giving both generic templates and pre-defined standard instances for numerical applications.

NAG ALGOL & ALGOL 68 Libraries. Two Program Libraries for the now near-defunct ALGOL and ALGOL 68 languages.

NAG SCS Ada Library. Complements the full Ada Library. Tailored to run on scalable systems.

NAG Pascal & Turbo Pascal Libraries. Procedures covering a wide range of problems in numerical computation. Available on PCs and mainframes.

NAG Graphics Library. A FORTRAN subroutine Library designed to complement the maths libraries. Contains display and plotting facilities with emphasis on the accuracy of representation.

NAH Help System. An interactive help system to guide the NAG FORTRAN Library user in the choice and use of individual subroutines.

NAG Workstation FORTRAN Library. Numerical and statistical subroutines chosen from the main library for use on PCs and small workstations.

NAG/SERC Finite Element Library. A collection of FORTRAN subroutines and template programs for solving finite element problems.

Harwell Sparse Matrix Library. A collection of FORTRAN subroutines for efficient matrix operations on sparse matrices.

DASL. Produced by the National Physical Laboratory. The Data Approximation Subroutine Library for fitting data onto two or more variables.

SLICOT. FORTRAN subroutine Library. Containing routines for control engineering applications.

Prices for the libraries vary by target platform. As a guide, the cheapest software is for PCs, and costs a few hundred pounds.

As already mentioned, our software is portable across a wide range of hardware. To complement this portability, we also offer a service of checking the accuracy of the arithmetic on a client's machine. It is worth noting that the results given by some floating point processors can be dangerously inaccurate.

Famines, floods and accountants

NAG software is used in a wide range of applications, including optimisation of food distribution in famine regions, study of airborne pollutants, satellite image processing, market research, aircraft design, financial portfolio management and weather prediction. It is impossible to define a typical user or a typical use of our libraries, so here are a couple of examples to illustrate the diversity.

The optimisation routines (Chapter E04) have been used to perform the risk analysis for share portfolios in the City. Our client needed to be able to quote interest rates for short term loans, given certain fixed rates for fixed periods, the spot rates. They wanted to do better than the linear interpolation that is generally used for this purpose. Straightforward polynomial or spline interpolation gives unwanted and highly undesirable fluctuations in the interpolant. However, using other E02 routines it was able to construct a solution that it believes gives it a competitive advantage.

I can't tell you much about the risk analysis package, for obvious commercial reasons. A more complete example can be given in the case of a chemical company, who used our simultaneous linear equation solvers combined with the matrix decomposition routines to determine the correct proportion of overhead (fixed costs) to be allocated to each part of a chemical process in which the intermediate chemical products were sold as products in their own right. For example, ethylene and chlorine combine to make a Vinyl Chloride monomer which is both sold on and used to produce the product PVC. A network of allocated costs can be constructed. Each internal cost centre has prime cost allocated to it and a set of fractional flows into the other cost centres, in effect a feedback loop. There are internal and external costs flows corre-

sponding to further production and sales respectively. This is fairly hairy stuff, as you can see by glancing at the following equations. The internal costs can be calculated as:

$Ci = C + CiAi$
and, therefore,
$Ci = C(In - Ai)^{-1}$
The external costs are
$Ce = C(In - Ai)^{-1} Ae$
then
$X = (In - Ai)^{-1} Ae$

where n is the number of internal centres, m the number of external centres, p the number of types of cost, Ai is an n by n matrix of fractional internal flows, Ae is an n by m matrix of fractional external costs, C is a p by n matrix of prime costs, Ce is a p by m matrix of external (allocated) costs, Ci is a p by n matrix of internal costs, In is the identity matrix and X is an *n by m resolved allocation matrix. Whew!*

Using the NAG routines, the internal costs Ci and the resolved allocation matrix X could be solved by calculating the inverse of the large, sparse matrix *(I-Ai)* using the specialist sparse matrix library. However, if you only had the plain vanilla FORTRAN Library (for example), the solution could be computed by reformulating the equations as:

$Ci(I-Ai) = C$
and
$(I-Ai)X = Ae$

The results given by some floating point processors can be dangerously inaccurate

Subroutine F01BRF can be used to obtain the LU decomposition of *(I-Ai)*, giving the upper and lower triangular matrices. These are then processed by F04AXF which is used once for each type of cost. With these routines, the costs on 1000 or more centres can be allocated, with the precise figure depending on the density of the matrix of internal flows.

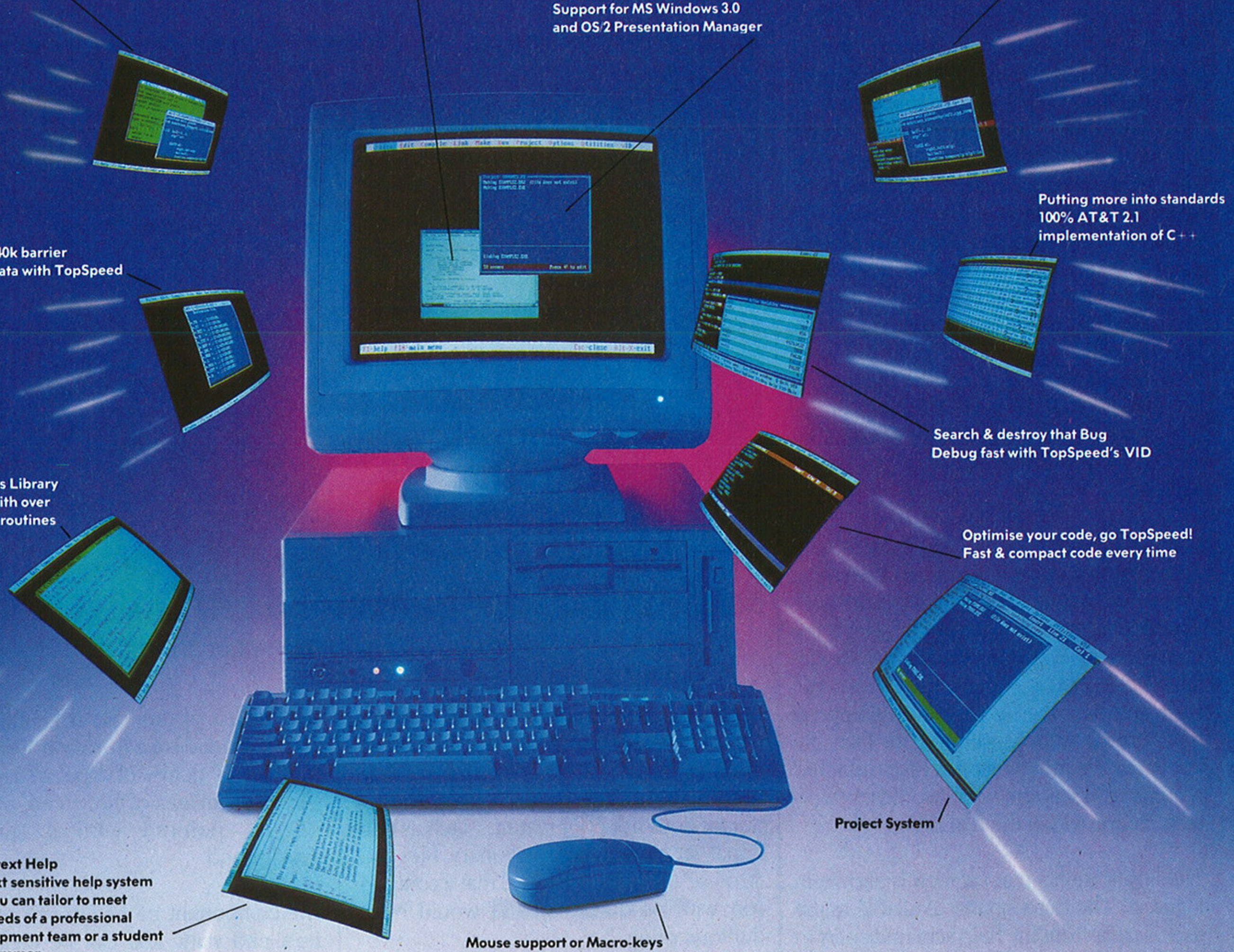
Rallying cry

Computing hardware is evolving at a pace that can make non-portable software obsolete in a very short period of time. Language extensions and new languages are not the complete answer, the user needs 'bigger building blocks' together with true portability across hardware models and hardware types whilst retaining performance. At NAG Ltd we are investing a lot of time and effort in producing software that meets the often conflicting goals of portability and performance, working on independent and collaborative projects with companies and Universities throughout the world. We are making the effort so that you don't have to!

EXE

Peter Anderton is the Marketing Manager for the Numerical Algorithms Group Ltd which can be reached in Oxford on 0865 511 245.

If you have to get more out of your next C++ Compiler make sure there's a lot more in it



Choosing your next compiler is no mean feat. Whether you're just putting a toe in the water or about to embark on a major development project with C++, you'll need to know that the compiler you choose gives you the productive edge.

TopSpeed C++ packs more in

TopSpeed C++ packs more in, whether you are developing for DOS, Windows 3 or OS/2. Develop mean-and-lean code with TopSpeed's super-optimizing code generator, Smart-Linking and Project system. Then, add the most extensive C++ class Object library available and you too can now harness the real power of C++.

TopSpeed Putting more into standards

TopSpeed C++ not only complies with the industry standards offering 100% conformance to AT&T V2.1. but also packs in a range of extensions to take your programming into hyperspace!

TopSpeed just the environment for *C++*

To make the most out of C++ and the power of OOP you'll need the development environment that packs the most in. TopSpeed's unique Project system frees you to be creative whilst it manages file dependencies, compilation, assembly, linking and much more.

TopSpeed C++ in a family of thoroughbreds

C++ is just one of a family of four compilers that integrate seamlessly into the TopSpeed Mulit-language Development Environment. At £59 per language you can custom-build a development system that takes your programming TopSpeed!

To place your order for **TopSpeed C++** or for a comprehensive information pack, call JPI on +44 (0) 234 267500, or write to;

Jensen & Partners International

Jensen & Partners UK Ltd, 3 The Mansards, Bedford, MK40 2RX, England.

We speak your language

*ANSI X3. 159-1989 Programming Language C Compliant Certified Sept 1991 Certification No:BSI 0087 09/09.

CIRCLE NO. 562

Zortech and the database

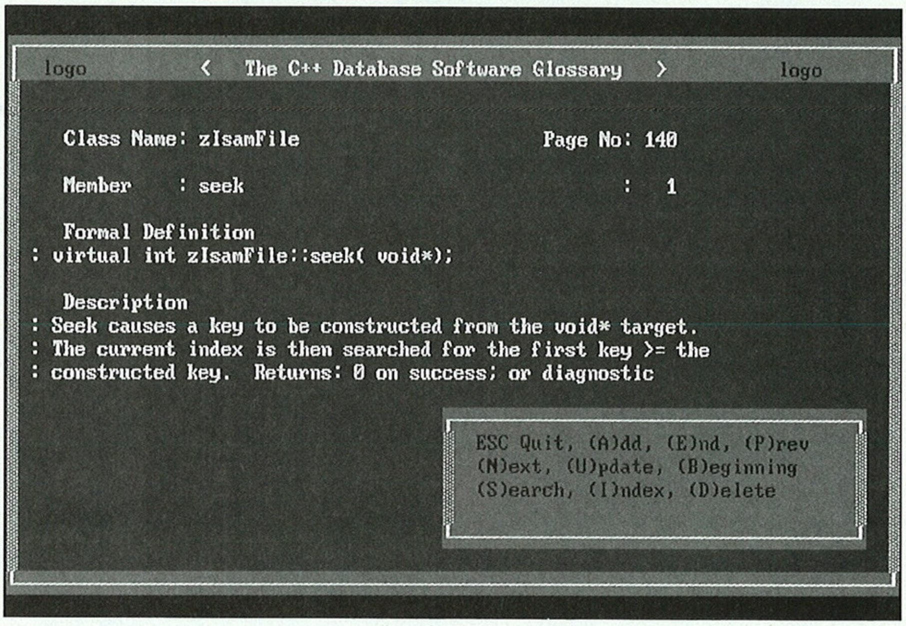
Zortech's C++ compilers are well-known, but what of the add-ons that the company offers? John Cant has been experimenting with the database library.

Zortech is best known for its range of C++ compilers; but it also distributes various C++ tools and libraries produced both by itself and third parties. The *C++ Database Library* is an in-house product which complements the V2.1 MS-DOS and OS/2 compilers. The library consists of two sets of classes. One contains the components needed to create an ISAM file system, the other consists of the fields and records required to create (text mode) interactive data entry screens. The package is distributed with fully-commented source code and can be compiled for single-user or multi-user applications. A screen generator utility is included to aid in the design of data entry screens.

Each component of the database library comes with an example program detailing its use, and a fully-fledged multi-user application is provided that shows off all the components. This consists of a flat file database designed to contain a glossary of the class and member names which comprise the Zortech database library. In fact, the application starts by filling in a few hundred records with random data - would that Zortech had provided the real thing.

The application's main screen is shown in Figure 1. The functionality available to the user is clear from the help screen display in the bottom right corner. My tests show that the program moves along nice and cleanly in a multi-user setup. However, if you try to expand this application, you encounter a disconcerting bug: the `glossary` structure is defined in two source files... differently.

According to the documentation, the library has been designed to achieve as much code-sharing as possible. Attention has been given to speed considerations, so that the modular hierarchical approach should not result in poor performance. A natural set of questions that are **not** dealt with in this review relate to performance. My initial intention was to compare the library with such products as dBASE IV, Paradox and the Paradox Engine library. As the Zortech database library does not offer features such as relational structure, data dictionary files and 'browse' facilities, I decided that a comparison with products that did would be of dubious value.



ADVANCED BUSINESS GRAPHICS

Figure 1 - The sample application

Structure

The approach used in the database library has been to build large structures out of small building blocks. The various classes are outlined in Figure 2. Although the components are designed with the aim of creating more complex structures, each can be used in its own right. For instance, the `zCache` class is used in an example program to implement a help-message system where the most-recently-used help messages are held in memory, with paging to and from disk.

The largest of the screen handling structures provided is a `zRecord`. This consists of a set of fields, of type `zField`, displayed on an area of the screen, `zScreenArea`, painted with a static text background.

The component `zFields`, `zScreenArea` and static text can be created by a screen generator utility ZSCRG. The range of fields provided includes a range of string fields (single character, masked and boolean), a 'NumericField' (`long`, `int`, `double` and `money`), and a 'DateField' (or USA or European).

`zRecord` is a fairly simple class that lends itself to being adapted to suit a particular application. For example, as it stands, the following all cause the program to exit the current field and progress to the next: UpArrow, DownArrow, PgUp, PgDn, Enter and mouse clicking. It is not hard to improve on this if required by one's application.

TopSpeed to CeBit'91 with JPI

WIN A TRIP TO HANNOVER

Occupying a site the size of a small town and spread over 23 halls each focussing on some particular aspect of computing and IT, CeBit this year runs for 7 days from March 13th to the 20th. JPI, (Jensen & Partners International), in association with .EXE magazine, is offering you the chance to win a trip to Europe's largest computing exhibition.

CeBit, always the scene for new product launches, will this year set the scene for the launch of TopSpeed version 3.0, a significant milestone in PC applications development. JPI, producers of TopSpeed C and TopSpeed Modula-2, will be showing version 3.0 of the TopSpeed integrated multi-language development environment. As well as being able to mix and match C & Modula-2 coded, JPI will add TopSpeed Pascal, (an ISO compliant Pascal with Objects and JPI extensions), and TopSpeed C++, (AT&T2.1) to the TopSpeed family.

The two first prize winners will each receive a ticket for their flight to Hanover. A limousine will take you direct to the fair for a full days browsing all that's new in computing. On return to the UK you will receive a complete TopSpeed Professional Developers Kit consisting of the TopSpeed integrated multi-language programming environment and all four TopSpeed languages. Winners may choose either TopSpeed for DOS or OS/2.

Ten runners up will each receive TopSpeed Environment and their choice of either TopSpeed C++, Pascal, Modula-2 or C for either DOS or OS/2. A further 15 runners up will receive a .EXE T-Shirt of their choice and a .EXE mug.

Simply answer the five following questions and send your answers by post to the address shown below:

Q1: Which is the only ANSI complaint C compiler for DOS based applications?

Q2: Which Modula-2 shares common pragma directives with three other TopSpeed languages?

Q3: What two languages are being added to the TopSpeed family of in integrated compilers?

Q4: Which compilers are brewed in England by Danes?

Q5: Which programming and systems journal celebrated its 5th year in (February) '91?

Name:.. Job Title ....

Organisation/company:..

Compiler:..

Primary language used: ....

Operating System:..

Address:..

Phone:.. Fax:..

Winners will be drawn from correct entries received by March 10th. Flight winners will be notified by fax the same day. All other prize winners will be notified by post. The judges decision is final.

Post or fax your answers to TopSpeed to CeBit'91. 3 The Mansards, Tavistock Street, Bedford, MK40 2RX. Fax +44 (0)234 217094

ZSCRG

To help you set up the data for these field and screen objects, Zortech supplies a screen generator. 'ZSCRG' operates in three phases. In phase one, you paint areas of the screen with boxes, colours and static text. Boxes are created using the extended IBM character set with a rather good 'turtle graphics' facility, putting in all the strange angles necessary to connect up the boxes.

In Phase 2 the data entry and display fields are defined. The cursor is placed where the field should start, and its various attributes are chosen from a menu. Each field can have its own associated help screen and verification function, and a variety of expected characteristics such as case conversion and necessary or defaulted input. The third phase lets you move and redefine the fields.

The user interface to ZSCRG is decidedly poor and non-standard; although with a bit of work it could become a powerful and useful program. Some examples: once you have created something, be careful to save it - Alt-X aborts and overwrites your original. The modification switch '-r', which is supposed to let you reposition existing fields, produces incomprehensible results.

The output is in the form of a C++ source file comprising a packed version of the static background, a function which creates all the specified fields, and prototypes for the associated help screens and verification functions. ZSCRG actually reads in the output file to let you alter your field definitions (as opposed to storing its own copy of the data separately in another file). It is pretty sensitive to hand modifications of this file. I found that the best policy was to save it somewhere safe and pull out the bits that you want to use. If, at some future date, you need to make changes using ZSCRG, you can fall back on the original generated file.

The user interface is decidedly poor, although with some work it could become a powerful program

A final limitation is that the screen colour attributes are hard-wired, so that it is non-trivial to support black-and-white monitors or dynamic colour changes.

ISAM system

The most complex structure that can be created with the file management system is an ISAM file. This consists of a structured data file together with associated indexes. The following functions can be performed on the file: `add()`, `change_index()`, `check()` (reports on integrity of index), `find()`, `first()`, `last()`, `next()`, `prev()`, `record_count()`, `refresh()`, `remove()`, `seek()` and `update()`.

Some fragments of code which illustrate the use of an ISAM class are given in Figure 3. First we have a definition of the entity `person` that is going to be stored in the ISAM. Then there is the definition of the keys that will index the file. An index can comprise a number of fragments, plus a method for comparing one set of fragments (key) with another. You also choose whether to allow duplicate keys.

The demo program creates an ISAM file with two indexes, and adds an instance of `person`, handling the return `DUPLICATE_KEY` as appropriate. It then switches to the second index and prints out a list of all the entries in the database.

Note that, as the system stands, there is no way to find out what information is stored in the file. You need to know independently

File system classes:

zPermanent	An abstract class to save objects to disk.
zFLRecFile	A derivative of `zPermanent` to implement fixed length data file systems.
zBtIndex	An implementation of Btree indexes using `zFLRecFile` as a Base class, and arbitrary key types.
zBtDupIndex	Btree indexes as above, but with duplicated keys allowed.
zIsamFile	Indexed sequential file access with multiple indexes and compound keys.
zVIsamFile	Indexed sequential file access as above, but with variable length data records.
zHeapFile	A derivative of `zPermanent` to implement variable length data file systems.
zCache	Least-recently-used cache implementation to minimise disk access.
zComparator	An abstract base class to implement arbitrarily complex comparisons between objects of the same type.
zGenComparator	Compares blocks of memor.y
zGenReverseComparator	Compares blocks of memory, working backwards through the block.
zIntegerComparator	Compares signed integers.
zCStringComparator	An interface to `strcmp()`.
zIsamComparator	Comparator used by `zIsamFile`.

Interactive screen classes:

zRectangle	An abstract base class useful for screen objects.
zScreenArea	Non-specific screen display - used to decouple data input/output from specific facilities.
zField	An abstract field class, from which the following classes are derived:
zStringField	With optional character filtering and case conversion.
zCharacterField	Special case of `zStringField`.
zBooleanField	Special case of `zStringField`.
zMaskedStringField	Semi-automatic input for structured strings.
zNumericField	An abstract class for numbers, from which are derived:
zLongField	Special case of `zNumericField`.
zIntField	Special case of `zNumericField`.
zDoubleField	Special case of `zNumericField`.
zMoneyField	Supports `zMoney` objects - see below.
zDateField	Dates in US or European forma.t
zMoney	General-purpose currency class, allegedly borrowed from Zortech's C++ tools.
zRecord	A generalisation of interactive data entry screens - a collection of fields.

```
zPermanent
├── zFLRecFile
│   ├── zBtIndex
│   │   └── zBtDupIndex
│   └── zIsamFile
│       └── zVIsamFile
└── zHeapFile

zCache

zComparator
├── zGenComparator
├── zGenReverseComparator
├── zIntegerComparator
├── zCStringComparator
└── zIsamComparator

zRectangle
└── zScreenArea

zField
├── zStringField
│   └── zCharacterField
│       └── zBooleanField
├── zMaskedStringField
├── zNumericField
│   ├── zLongField
│   │   └── zIntField
│   ├── zDoubleField
│   └── zMoneyField
└── zDateField

zMoney

zRecord
```

Figure 2 - Principle classes

that what is in there is a `struct person` indexed by `{KeySpec1, KeySpec2}`. Normally, database packages store this information in a separate data dictionary file or a special header; here, this would have to be generated by hand - no utility is provided.

Of course, the facility to simply stash items away in a structured file is very useful in many contexts, whether one is developing a database or not.

Multi-user applications

The file system library has been designed so that an application can be compiled as either single-user or as multi-user. Multi-user access to the file system is provided via protocol locking. Any transaction must check the 'lock count' before performing a write operation. All the entities associated with a transaction are gathered into a successfully locked group before any data is changed. If any required lock fails, all locks are released and a fresh attempt is made from scratch. All this is sped through by a caching system that attempts to keep as much current data in memory for as long as possible.

From the developer's point of view, dealing with multi-user transactions is surprisingly simple, as Figure 4 illustrates. In this example, if adding the record fails due to an inability to obtain a lock, the application waits for five time units (system dependent), and then retries.

The first release of the database is best passed over - although it did get me into the debugger

Documentation

Documentation consists of one 270 page perfect-bound manual which covers everything. The manual leaves a lot to be desired. The many typos and missing references in the index can be very frustrating. In a similar fashion to the other Zortech manuals, there are many references for a given topic - it would have been very helpful if one of these references was highlighted, earmarking it as the principal one.

For some reason, the function reference sections are organised with members of a particular class listed in non-alphabetical order (in fact, they are not listed in any particular order).

Furthermore, members from inherited classes are not documented at all. With many classes derived from others, you often have a selection of inherited members available to you. For example, the `zIsamFile.wait()` call of Figure 4 is not documented as a member of the `zIsamFile` class; it happens to be a member of the `zPermanent` class - `zIsamFile`'s base class. Some documentation method is needed to make it plain exactly what facilities are available to every class, and how the classes are related (the tree diagram in Figure 1 was produced especially for this article - it does not form part of Zortech's supplied documentation, nor is there anything like it).

The source files themselves are so fully commented that it is hard to see the code for the trees.

In real life

The first release of the database is best passed over without much comment. Demonstration programs crashed, ZSCRG was close to unusable. However, on the bright side, tracking down the faults in the initial release got me into Zortech's debugger, which is unequivocally astonishing, particularly when used in extended memory mode. (Incidentally, using the debugger is an excellent way to explore what C++ is up to under the surface - eg discovering exactly when the destructors get called. It is also worth running your final product user under the debugger and get it to watch out for un-deallocated memory - very illuminating!)

A typical problem was in the `zPermanent` class which saves objects to disk. If a file did not exist, then the constructor aborted without creating any of the `new` entities. When the destructor was called later, all the 'uncreated' entities were `deleted`, causing havoc.

Apart from the simple bugs already mentioned, I haven't found the need to delve into the depths of the ISAM file system - for my purposes, the functionality provided has been fine.

```
struct person
{
    char surname[20];
    char firstname[20];
    char city[20];
};

int KeySpec1[] =
{
     1,                              // one fragment
     1,                              // duplicates OK
     offsetof( person, surname[0]),  // first fragment offset
     20,                             // first fragment length
     0,                              // comparison method
                                     // (string comparison in this case) };

int KeySpec2[] = { 1, 1, offsetof( person, firstname[0]), 20, 0, };
int* KeySpecList[] = { KeySpec1, KeySpec2};

int rv;
person individual;

zIsamFile zIfile( "persons", 2, KeySpecList, sizeof( person));
if( (rv = zIfile.diagnostic()) != 0) Error( "create failed: %d", rv);

// prompt user for a "person"
GetPersonFromUser( &individual);

if( (rv = zIfile.add( &individual)) != 0)
{
     if( rv == DUPLICATE_KEY) Warning( "Duplicate Key");
     else Error( "Add failed: %d", rv);
}

// scan through the file system on second index
zIfile.change_index( 1, &individual);
zIfile.first( &individual);
do
{
     printf( "%s %s from %s\n",
             individual.firstname,
             individual.surname, individual.city);
} while( zIfile.next( &individual) == 0);

// check the file system is OK
zIfile.check( &individual, 1);
```

Figure 3 - Using ISAM

NEW VERSION 2!

NEW! WEITEK SUPPORT

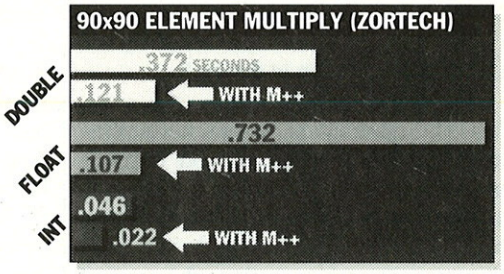
"Are you taking advantage of the tremendous power and flexibility of C++?"

The C++ language is an incredible language for modern program development. C++, unlike C, was designed to be extended. **M++. Unleash the Power of C++.**

The powerful array handling capabilities that have accelerated development in advanced scientific languages for years are now available in C++. **M++.** The M++ library is a complete, standardized, multidimensional array language extension to C++. With M++, your memory and array handling problems are gone. Instead, you have a powerful environment for developing tight, fast applications from the latest graphics to advanced numerics. Use the M++ classes directly or as a critical solid base for your C++ class development. No Royalties! Beyond a matrix language, M++ is an object oriented, multidimensional, array language extension to C++ for statistical, scientific and engineering program development.

New! M++ Modules: M++SUM, M++LSM, and M++OPTIM

The new M++ modules extend the M++ library to provide state-of-the-art statistical, generalized least squares, and optimization classes for scientific, statistical and engineering programmers. The M++ modules further extend C++ into an advanced, object-oriented statistical or scientific programming environment.



16 MHz 386 with 387

SUM-Statistical Utilities Module - scalar and array versions of functions for computing normal, t, and F cdf's, incomplete gamma and beta integrals, and other special functions.

LSM-Least Squares Module - linear and nonlinear, single and multiple equation least squares estimation.

OPTIM - General quasi-Newton and Newton-Raphson optimization with various step length methods.

SPECIAL EUROPEAN INTRODUCTORY OFFER

M++ WITH SOURCE

£199.95

081-316-7777

ZORTECH LIMITED
58-60 BERESFORD STREET, LONDON SE18 6BG

dyad
SOFTWARE CORPORATION
16950 151st Avenue Southeast, Renton, WA, 98058-8627 (206) 255-0312 FAX: (206) 228-3178

MICROSOFT WINDOWS

	DOS/Windows OS/2	Sun/Unix
M++ Library	$295/$495 *	$495/$695 *
M++ SUM	$95 *	$135 *
M++ LSM	$135/$195 *	$175/$245 *
M++OPTIM	$135/$195 *	$175/$245 *
M++ LP	$135/$195 *	$175/$245 *

* Includes complete source code. Call for pricing on 386 Unix and other systems. Compatible with all AT&T 2.0+ C++ compilers including Zortech, Borland, GNU, and Glockenspiel.

CIRCLE NO. 567

```
    // attempt to add a record
    for( ;;)
    {
        int rv = zIfile.add( &individual);
        if( !rv) break;
#ifdef MU
        else if( rv == LOCKFAIL)
        {
            zIfile.wait( 5);
            continue;
        }
#endif
        else Error( "Add error %d", rv);
    }
```

Figure 4 - Multi-user capabilities

I have spent quite a bit of time, however, customising the `zRecord` class and, in retrospect, I feel that it has been rather poorly designed.

The definition of a `zRecord` requires the following parameters: two `zScreenArea*`s, a `StaticText*`, a `zField**` and a `void*` object that is to be filled in. In the supplied example program, the `zRecord` and the `zField**` are automatics. In building a more complex application, I soon needed to declare a `zRecord*`, and create the actual record using `new`, as shown in Figure 5.

This led to intermittent crashes, not surprisingly because the `zfps` went out of scope on exit from `OpenRecord()`. All very easy to see with hindsight, but I would say that this is a bad design of the record class. All important entities, such as the array of fields, should be carried around by the class in private.

In addition, you are liable to forget that the '`zfps`' have been assigned to `new` fields in another function, and they are likely not to get deleted. The designers of the database themselves fell into this trap. Running the sample program GLOSSARY through the Zortech debugger heap checking facility had the program exiting without de-allocating the memory for these fields. This lack of care in de-allocating memory also crops up in Zortech's Tools Library - in the text editor `ztext`, for example.

While I'm on the subject of the Tools, the `zScreenArea` class used by the database library is not the same as the class of the same name in the Tools library. To avoid linker clashes, one must remove the 'screen' and 'money' classes from the database library, as well as the associated `#include` files. Also, the database `#include` files are not truly compatible with the tools `#includes` - in particular, various keyboard definitions are doubly defined; this takes a bit of sorting out.

A number of other bugs are detailed on the Zortech bulletin board.

There is also a speed consideration with regard to the design of the `zRecord` class. As it stands, the generated function that makes the fields creates all the `new` entities before throwing up the screen. Even on a fast system this leads to an appreciable delay in flipping between screens.

To date, I have received two free upgrades of the database. While such support is excellent, the upgrades have splattered on top of the existing version. If the database is out of synch with the Tools library and with your own source code, nightmares follow as surely as if you had spent the evening punishing the Lymeswold. For instance, in the latest release many `ints` had been changed to `shorts`. The lesson is - never throw away an old version until you are sure that the new is compatible with the rest of your environment.

```
void UseRecord( zScreenArea* ScreenArea,
                const short* StaticText,
                int NumFields,
                void* target,
                zScreenArea* HelpScreenArea)
{
  zField* zfp[];
  zRecord zRec( ScreenArea, StaticText, NumFields, zfp,
                target, HelpScreenArea);
}

becomes:

zRecord* zRecPtr;

void OpenRecord( zScreenArea* ScreenArea,
                 const short* StaticText,
                 int NumFields,
                 void* target,
                 zScreenArea* HelpScreenArea)
{
  zField* zfp[];
  zRecPtr = new zRecord( ScreenArea, StaticText,
                         NumFields, zfp,
                         target, HelpScreenArea);
}

void CloseRecord()
{
  delete zRecPtr;
}
```

Figure 5 - zRecord design problems

Conclusion

Struggling through the early releases of this product, trying to distinguish bugs in the libraries from bugs in my application, it became clear to me that classes are inherently more complex than the ordinary library functions. The provision of source code and a source-code debugger is now a necessity, not a luxury. Zortech has taken a courageous and laudable stand in providing source code for its function and class libraries.

In the best of all possible worlds, the classes would be provided with built-in `#ifdef DEBUG` logging-to-file and tracing help. This is a tall order, but the complexity of these tools is such that I have spent a lot of time reaching for the big red switch, and would give anything for a pointer to something other than illegal memory.

The releases to date of both the tools and database libraries have contained so many trivial bugs that I am left wondering about Zortech's beta-test procedures. I have the uneasy feeling that, if simple things like opening files are not handled properly, little care can have been taken elsewhere in the architecture.

However, a great deal of effort has gone in to designing a highly valuable toolbox of classes which, once the initial teething problems are ironed out, will be of great value in themselves and as building blocks for larger structures. While Zortech may be accused of having rushed out a product that needed more rigorous testing, its bulletin board help line is excellent: responsive and capable.

EXE

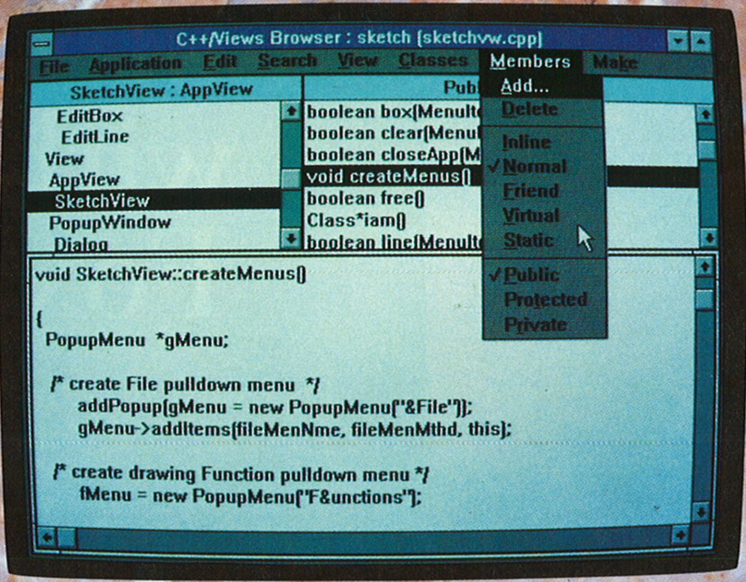
John Cant has been an independent software consultant for the last eight years, specialising in support for scientific research programmes and communications using C, C++ and assembly languages. John is an associate of PHD Computer Consultants (033 488 417).

Zortech's C++ Database Library V2.1 is available direct from Zortech Ltd (081 316 7777) at a cost of £200.

C++/Views™

for Microsoft Windows

SPECIAL
INTRODUCTORY
EUROPEAN PRICING
AVAILABLE FROM ZORTECH !
£200 (+VAT)
CALL 081 316 7777



THE MICROSOFT WINDOWS 3.0 DEVELOPMENT TOOL THAT DELIVERS FROM START TO FINISH.

C++/Views is a development tool for C++ programmers that not only reduces the complexity of Microsoft Windows **3.0** but also slashes development time by up to 75%.

Delivers on the promise of Object-Oriented Programming (OOP).
Encapsulates more MS Windows 3.0 functionality than any other tool on the market today. Get MS Windows applications off to a fast start with a framework of over 65 tested and ready-to-go C++ classes.

Has the most complete C++ class library for MS Windows Development.
Get started with graphical user interface classes such as windows, views, bitmaps, dialog boxes, menus, popup menus, graphics, regions, pens, brushes, controls, buttons, listboxes, valuators, editors, printers and much more. Organize your data with foundation classes such as containers, collections, sets, dictionaries, files, strings, streams and so-on. Use other classes to manage the persistance of objects across files, to perform serial communications, and to activate timed events.

Provides support for the entire project.
Comes with a complete OOP development environment including the first fully functional C++ class hierarchy *Browser.* Also includes an *Interface Generator* for building C++ dialog classes and a *Documentor* for automatically producing high quality documentation of your classes.

Works with Zortech C++.
Combine C++/Views with the **Zortech C++** compiler for a cost-effective and highly productive development environment for building your next generation software systems.

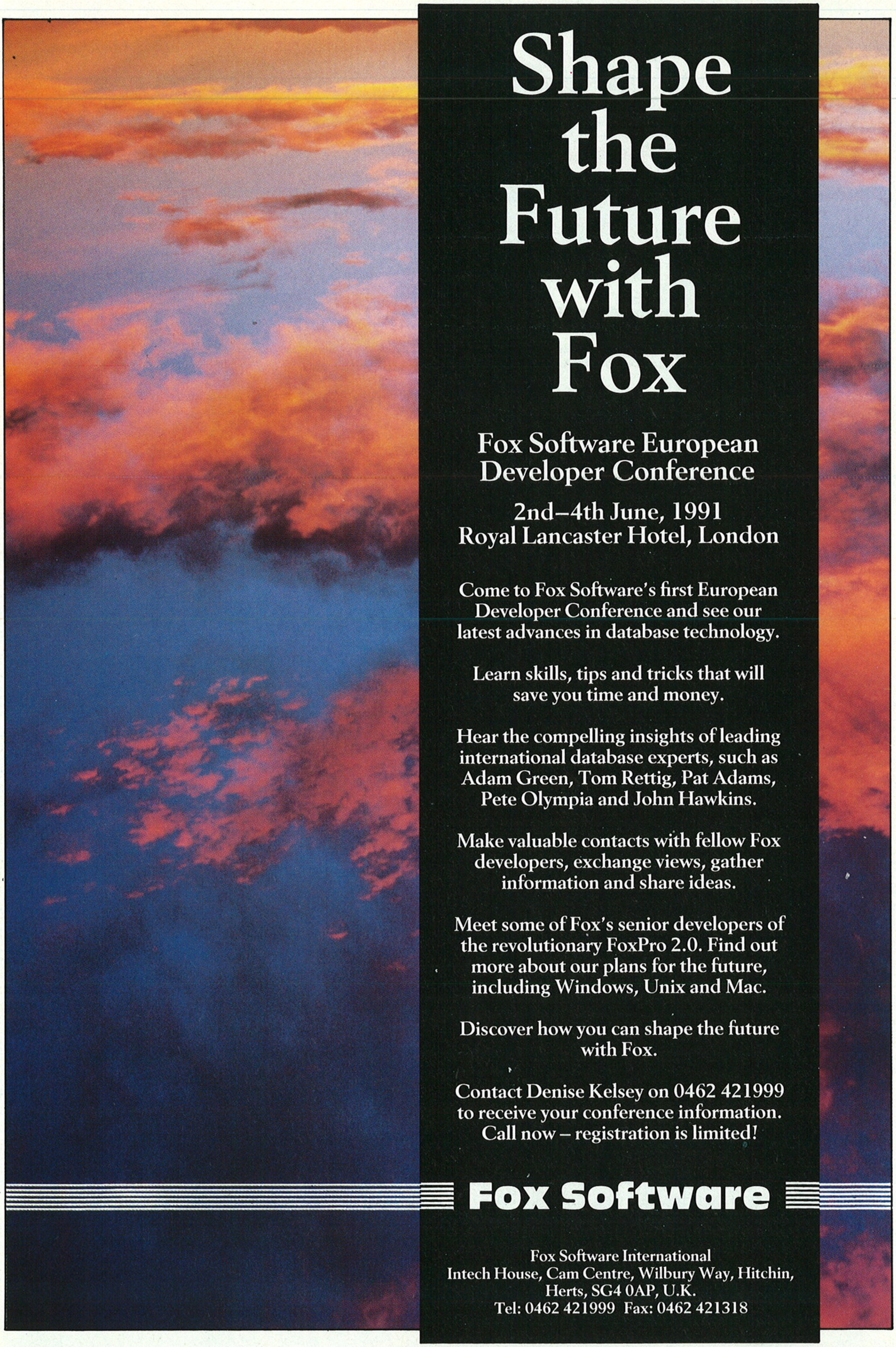
Pays for itself on even the smallest project.
Only **$495.00** with no royalties.
Comes complete with source code.

C++/Views
from CNS, Inc.

CNS, Inc., Software Products Dept.
7090 Shady Oak Road, Minneapolis, MN 55344
(612)-944-0170 • FAX (612)-944-0923

MICROSOFT WINDOWS

CIRCLE NO. 568



CIRCLE NO. 569

Turbo in a suit

Borland's new compiler sports a new name, a new environment (Windows) - and a new price. Dan O'Brien weighs up the changes.

First, the marketing. Borland C++ V2.00 is, by any other name, the Windows upgrade to Turbo C++ Professional V1.00. It seems that Borland feel that some buyers still see the old monicker as a 'cheap and cheerful' compiler tag. And as its new product is clearly pitched, in price (£300.00) and advertising, as a replacement for the Microsoft C V6.0 and the Windows SDK, this has become a unwelcome train of customer thought. Whatever the reason, it does prompt the fear that Borland is, as it distances itself from the pricing of the Turbo software, also dismissing the 'decent compilers for all' message of those packages. Could Borland C++ be another product on the compiler market priced on its corporate clout rather than features? Has Borland, we dare to ask, sold out?

Well, to be fair, what Borland C++ is not, is a cynical repackaging of an old compiler. A look through the package shows that there has been an extensive reworking of all the core programs. Out of the Turbo C++ Professional bundle of compiler, assembler, debugger and profiler, only the profiler has remained relatively untouched. The biggest modification, of course, is that the others now allow Windows programming - no easy task, involving new libraries, new headers, a considerably extended segmented linker and assembler, and even additional C++ keywords. New third party utilities have been bought in to support the package. Turbo Debugger now allows programmers to debug Windows applications and DLLs without the need for a second monitor.

Borland C++ is not just a cynical repackaging of an old compiler

Borland's compilers and assembler now both have protected mode versions. Even the editor has been tweaked, offering a new (and much wished for) undo and redo facility. All very encouraging. But does it work?

Getting It To Work

Getting it to work, frankly, needs a forklift truck. Buyers are supplied with eleven 3½" disks, seven 5¼" disks, seven manuals, and the usual wad of Borland promotional miscellanea. Installation is predictably long-winded but unfussy, with about 20 minutes of mindless disk shuffling required. Fully installed with sample code and all models catered for, Borland C++ takes up around 15 MB of hard disk space. This, thankfully, can be trimmed through diligent choice of models and so on. However, Borland C++, like its Turbo predecessor, still leaves the task of calculating the size of the different options to the user. The little 'Installation Complete' box at the end of the road includes a gleeful note to ex-Microsoft C users, welcoming them and reminding them of the User Guide chapter devoted to their acclimatisation. This is the first of many indications that Borland intends this Windows compiler to be not just an accompaniment to the Microsoft development products, but a complete replacement.

	Using MAKE	In the environment	Size
Zortech C++ V2.17/LINK	87.8	91	18064
Microsoft C V6.00/LINK	123.1	130	15504
Borland C++ V2.00/TLINK	47.7	129	24064
Borland C++ (precompiled headers)	30.9	88	24064
Borland C++ (Turbo Drive,headers)	38.5	59	24064

The application used was an adapted version of Multipad from the Microsoft SDK.
The system was a 25Mhz 386.
Times are in secs and include all linking and compilation, excluding initial RC compilation
Size of source is in bytes. Default optimisations space where used.

Table 1 - Compiling Windows Applications

Borland and Friends

To explain why this is so revolutionary requires a brief recap on the contents of the Microsoft kit. In the SDK are three modules without which Windows compilation is impossible. There is the set of essential - and copyrighted - library and header code. It contains (or used to contain: Microsoft now market it separately) Microsoft's segmented linker which is able to produce the extended .EXE format for Windows memory and dynamic library management. Third, it contains the Resource Compiler; the only program capable of turning definitions of dialogue boxes, menus, and graphics into a form digestible by Windows. And on top of all this are the SDK utilities: dialogue, icon and bitmap designing tools, run-time memory and message examiners, a font editor, help compiler and a posse of profilers and debuggers. In order to market a Windows compiler which could work without the SDK, a company would have not only to re-engineer much of the above: it would have to surmount Microsoft's copyright on the irreducible stuff, like the (vast) Windows `#INCLUDE` files.

Borland has managed to get around these problems, though not without some help. The new library code was written in-house, as were suitable linkers and debuggers. Whitewater sell a competitor to the SDK resource design tools, and Borland has bought this in. Whitewater's Resource Toolkit can also save completed resources directly into a .EXE file, dismissing the need for a separate resource compiler. Nevertheless, in order to remain completely compatible with code written with the SDK, Borland has chosen to license from Microsoft both the SDK header files and the Resource Compiler utility. Deep in the heart of this giant-killer, then, lie some suspiciously extra-large garments. Nonetheless, with little honour lost, Borland now has what was previously presumed to be impossible: a non-Microsoft off-the-shelf C for Windows.

Beside the BC IDE

Like any Borland language, this compiler is run from the Integrated Development Environment. Borland C++ has a choice of two: BC, and BCX, the DOS extended Version. The DOS extender itself is a separate program, again bought in by Borland, called Turbo Drive. This can be pre-loaded at the AUTOEXEC.BAT stage to save time and bother. The chief advantage of BCX is a generally faster compile and link turnaround and, of course, larger source handling. The downside is that screen updating and disk access are noticeably slower, making editing a tad less responsive. All but the smallest Window programs will need the Turbo Driven compiler. Turbo Drive uses the VCPI protocol, so it can run in Standard but not Enhanced mode. Writers of Enhanced applications will have to reboot Windows to test their software. Thus Borland C++ still cannot quite claim to be a development system running entirely within Windows. A shame, and one that I sincerely hope Borland will correct once the DPMI protocol becomes available.

Borland appears, in its old age, to have settled on a corporate look to its CUIs. Love it or hate it, the Borland C++ interface is now the same as Turbo Debugger, Turbo Profiler, Turbo Pascal, any program developed under Turbo Pascal's Turbo Vision class library, and even the new SideKick. For Borland C++ users, the most obvious editor boon is the inclusion of a Brief-like undo/redo facility, allowing up to 64 KB of alterations to a source file to be spurned and reinstated as desired. To be picky, however, the editor does still suffer from the features which irritated Turbo C++ users. The new editor is still, in places, a rather confused hybrid of CUA convention and Borland tradition. For example, Shift-Delete still cuts to the clipboard, Shift-Insert pastes, and Ctrl-Insert copies - ie it's a non-destructive cut. So, under this CUA logic, Ctrl-Delete, if it did anything, should delete the clipboard while preserving the blocked text - a non-destructive paste, as it were.

I can't imagine Turbo Debugger remaining the only decent Windows debugger for long

Actually, it deletes your block and preserves the buffer. Very confusing. Another quibble is the menu handler which insists on returning you to the editor rather than the parent menu if you press the <OK> or <CANCEL> buttons inside a sub-menu. Again, this would be perfectly acceptable on menus that were designed for this, but Borland C++ inherits the Turbo tradition menu structure, which was not. Thus the (quite common) move from sub-menu Options/ Compiler/ Code Generation to Options/ Compiler/ Optimisation is now a mess of detours.

Apart from Undo, and these niggles, there is little visible change to the IDE. Borland C++ inherits Turbo C++'s excellent linking to external applications, put to good use with the complicated set of utilities required for Windows compilation. All external program output is redirected into a special window, and any errors (including those from Microsoft's Resource Compiler) are converted into a standard format message, together with line number cross-reference, and accompanying help hypertext. Borland's project manager (the in-editor MAKE) has become even more sophisticated: as Figure 3 shows, compiling a Windows application (or indeed any other multiple file project) is now just a matter of adding the files required to a project at the bottom of the screen. One key press sorts out the intricacies of file dependency and differing source code requirements. So, if a project consists of a DLL .LIB file, Resource Script, two C++ files and an assembler module, the IDE will correctly detect which files require recompilation, and dole them out to the correct application. Choosing the form of the executable is just as easy. A new dialog box offers a simple choice of four: Standard DOS Application, DOS Overlay (Borland's VROOMM 'technology'), Windows Application, or Windows DLL. Any selection will automatically set up reasonable defaults, and automatically link in suitable start-up code. The IDE, as ever, has a fully featured ANSI-ish (five Plum Hall Sampler tests failed) C and C++ compiler built in, and now also sports a (limited) built in assembler, forgoing the old need to shell out to TASM. UNIX and DOS die-hards can use the command line version of the compiler with Borland's decent MAKE implementation. A protected mode command line compiler is provided.

Windows in Detail

Innocuous changes in the outward interface hide some pretty considerable changes underneath. Borland has evidently taken a great deal of care in its implementation of Windows compatibility, and it shows. Apart from the sheer fact it works at all, their are a number of pleasant tweaks which demonstrate an attention to detail and programmer comfort not always found in compiler technology.

The biggest boon to Windows programmers has to be the pre-compiled headers option. As far as I know, this feature is unique to Borland C++ in the DOS world, although a number of Macintosh compilers implement it. Simply put, Borland C++ can now dump the compiler symbol table to disk after compiling a set of `#INCLUDE` files. If another source file begins with the same set of inclusions, it loads up the raw symbol data instead of re-parsing the original file. This gives a speed improvement (says Borland) of around tenfold. Before you wave goodbye to 'Compiling WINDOWS.H' messages, however, there are some provisos. `INCLUDE` files must be organised in exactly the same order in every source file for Borland C++ to notice the similarity. There are a whole batch of options - memory model, target environment etc - which must be the same for pre-compilation to operate. The process has limited fine-tuning - you can't extract symbol dumps for one include file and move it to another project, for example. Symbol dumping can take up a great deal of disc space, and conditional compilation will naturally require re-parsing. Nonetheless, as our tests show for programs that usually require the re-compilation of the 120 KB WINDOWS.H for every single source file, the speed improvement is significant. And the pre-compiler can cope with class and function prototypes, including inline functions, so C++ applications should show a considerable decrease in compilation time, too.

When users aren't sure what they want, you could do with a crystal ball.

Let's face it, we all have to deal with users having only a hazy idea of what they want from an Expert System. Not surprisingly, it leaves you and your team uncertain about exactly what needs to be produced.

Fortunately XpertRule will help you to read their minds.

Whether you want to build a complex Expert System or simply add intelligent rule-based modules to a current application, XpertRule is an indispensible part of your toolkit. It includes powerful knowledge structures and rule induction facilities, enabling rule-based knowledge to be represented, captured and tested with ease. Once captured, the rules are generated in a choice of high level languages.

And knowledge-based applications developed with XpertRule can be ported across different hardware platforms — PC, mini or mainframe — since the generated "source code can be compiled for a range of operating systems.

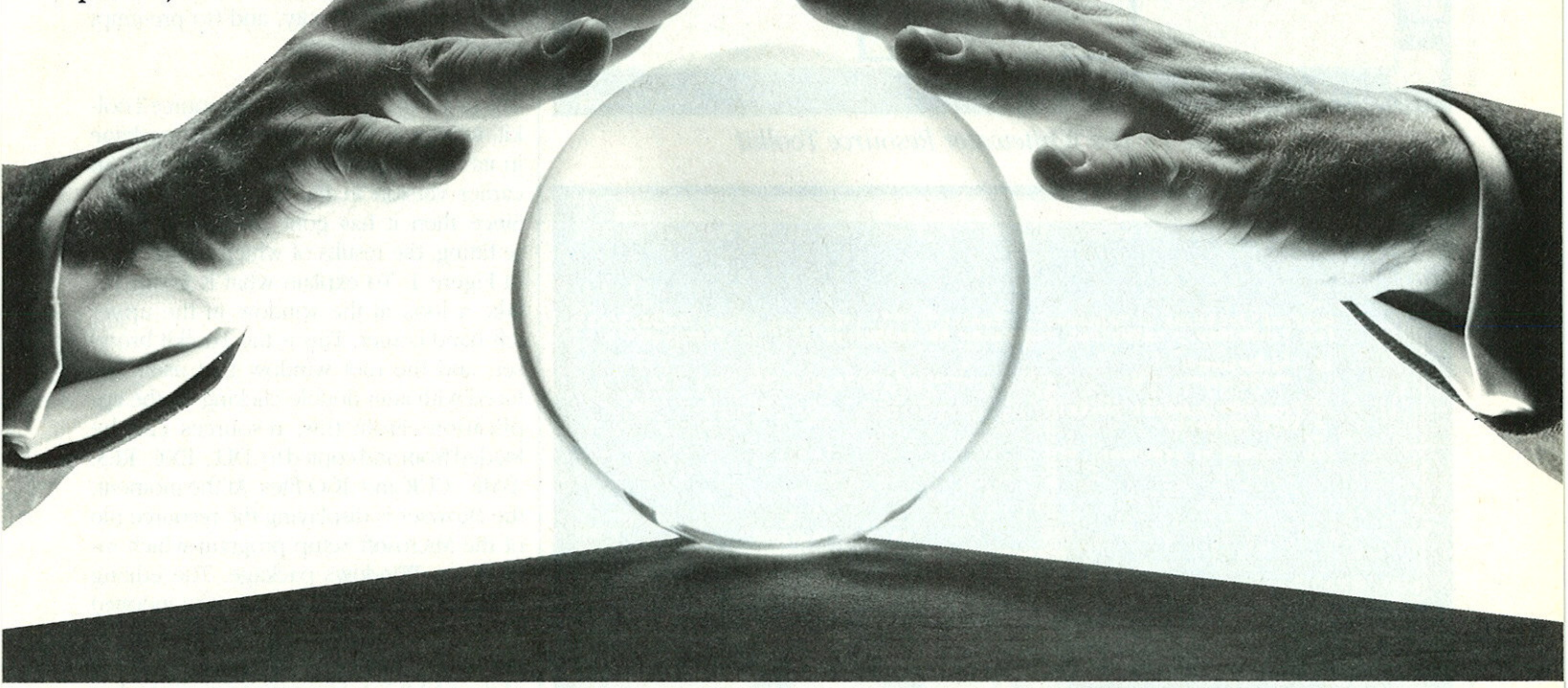
Unlike other Expert Systems, there's no need to learn a new procedural language either.

XpertRule runs on any IBM PC/XT/AT or compatible, under PC-DOS or MS-DOS 2.1 or higher with a minimum of 256K Ram (640K recommended) and a hard disk drive. It can generate code for Pascal, C, Cobol and RPG/400, and also has interfaces to Crystal and the Aion Development System (ADS) expert system shells.

In the future you'll see the XpertRule has helped you to build better applications much faster. Look into it.

For more information call Chris Wheeler on 0462 422525 or contact an approved dealer shown below.

XpertRule worth looking into.



APPROVED DEALERS

Absolute Computing Limited, Absolute House
124 Leavesden Road, Watford, Herts. WD2 5EG. Tel: (0923) 242570

Computer Solutions Limited, Canada Road, Byfleet
Surrey KT14 7HQ. Tel: (09323) 52744

Grey Matter Limited, Prigg Meadow, Ashburton
Devon TQ13 7DF. Tel: (0364) 53499

Roundhill Computer Systems Limited, Orchard House
Ogbourne St. George, Marlborough, Wiltshire SN8 1SU. Tel: (0672) 84535

Reflex Technology Limited, 9 Buckingham Place
Bellfield Road, High Wycombe, Bucks HP13 5HW. Tel: (0494) 465907

Frontline Distribution Limited, Intec 1, Wade Road
Basingstoke, Hampshire RG24 0NE. Tel: (0256) 463344

System Science, 3-5 Cynthia Street
London N1 9JF. Tel: 071-833 1022

Software Express, Portmill House, Portmill Lane
Hitchin, Herts SG5 1DJ. Tel: (0462) 422525

Software Generation · Portmill House, Portmill Lane, Hitchin, Herts SG5 1DJ

· DEALER ENQUIRIES WELCOME ·

CIRCLE NO. 570

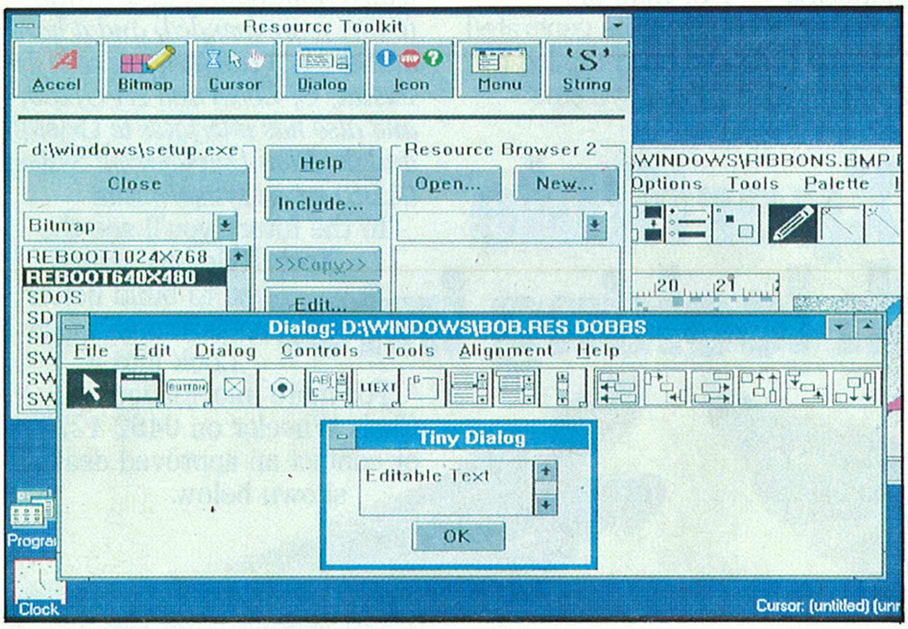
There are other, less obvious improvements too. Borland has implemented a very intelligent way of dealing with Windows callback functions, for instance. In order to link properly with Windows, callbacks have traditionally had to be explicitly listed in the linker .DEF file, and also surrounded with special entry and exit code when being used as a callback, so that they always refer to the correct data segment. Maintaining .DEF files can be a pain, and the Microsoft entry code requires most callback functions to be bound to their data segment at run-time, using `MakeProcInstance()`. Borland use two techniques to overcome these problems. The first is a new type modifier, `_export`, which removes the need for a separate .DEF file maintenance. More ingeniously, Borland has provided what it calls Smart Callbacks where, rather than being given the value of the DS register by Windows, exported functions simply copy it from the stack segment register. Obviously, this only works in memory models where the stack segment and the data segment are the same, but in these cases, the advantages are numerous: it's marginally faster, and because it works whatever is calling the procedure, you don't need to use `MakeProcInstance`, nor explicitly export the function. So you use `_export` when you need to be portable (and explicit) about such things and smart callbacks when you just need to knock off a quick executable. And `_export` has another use, too: if you use it to modify a class declaration in a DLL, you automatically export all of its non-inline member functions and static data members, too. The alternative would be to painstakingly enter in the mangled names into .DEF files. Borland C++ also provides a `huge` modifier to class declarations, forcing `this` and virtual table pointers to be full 32-bit pointers. These two features combined make implementing dynamic class libraries far more feasible than in Zortech C++, Borland's current competitor in this field.

Can Borland C++ do no wrong? Well, Borlands' code optimisation is still not up to much, with Zortech and Microsoft both well ahead. And while any Windows compilation cleverness doesn't slow down any DOS compilations you may do, the new segmented linker does. And perhaps Borland should not be let off so lightly for introducing new keywords into an already complicated environment. But then, Borland has long since stopped selling because it was the quickest or the most compact, or the most conformant code producer in the marketplace. Borland now trade on the sheer convenience of its products - and this is certainly a convenient compiler to use.

But there's more...

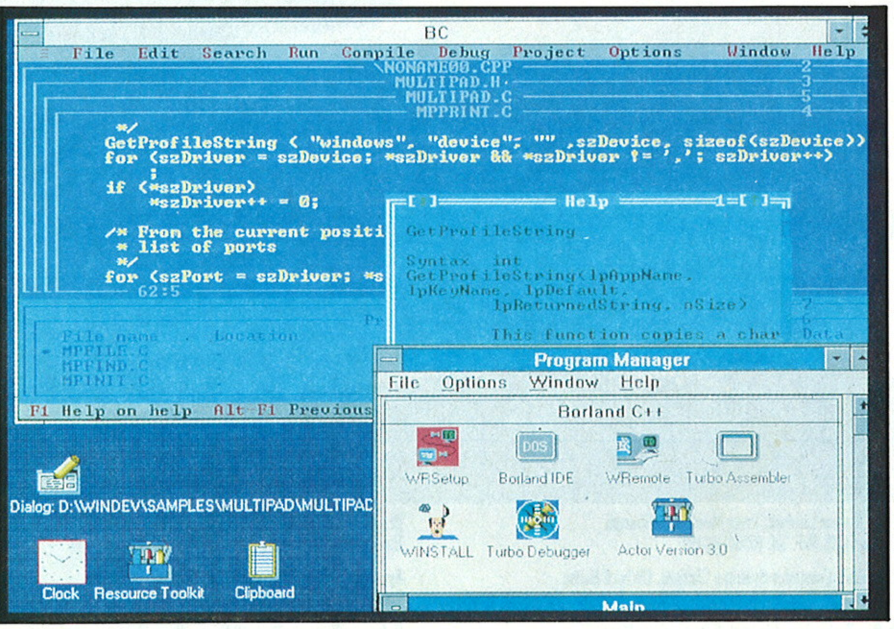
And that would be that, if Borland C++ was to be sold as just a compiler. But it would be fair to say that this package will sell just as much on the bundled utilities as it will on the C++ compiler. There are at least two more programs in the box which match the compiler in complexity, and (to pre-empt myself) impressiveness.

The first is the Whitewater's Resource Toolkit, written in Actor. Still sold as a package in its own right, Dave Jewell reviewed an earlier version in the .EXE of May 1990. Since then it has gone through a major re-fitting, the results of which can be seen in Figure 1. To explain what is going on, take a look at the window in the upper left-hand corner. This is the Toolkit browser, and the root window that users are faced with after double-clicking on the application. From this, resources can be loaded from and copied to .DLL, .EXE, .RES, .BMP, .CUR and .ICO files. At the moment, the Browser is displaying the resourcé file of the Microsoft Setup program which installs the Windows package. The editing utilities are spawned (as separate windows) on clicking one of the seven buttons at the top of the browser. A bit against the spirit of the SAA/CUA conventions perhaps, but nonetheless pretty intuitive. Shown here are the dialogue editor (foreground), bit map editor (far background), and the cursor editor (iconised, right).



ADVANCED BUSINESS GRAPHICS

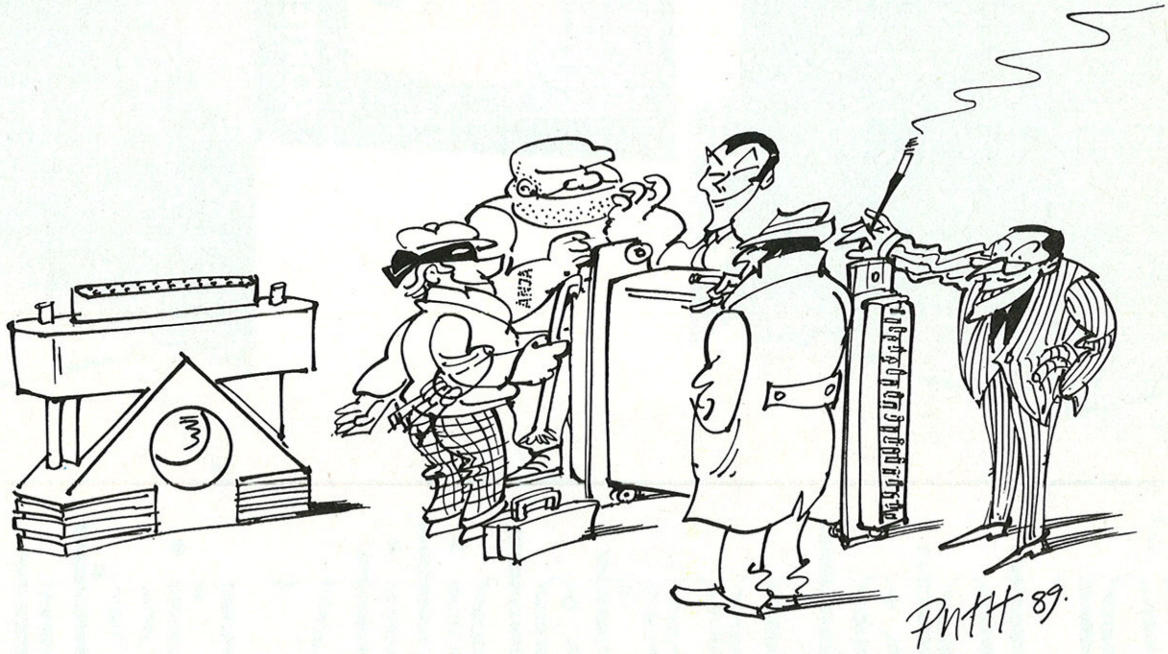
Figure 1 - The Whitewater Resource Toolkit



ADVANCED BUSINESS GRAPHICS

Figure 2 - The Package under Windows

5 out of 5 hackers prefer other software protection methods to Hardlock E-Y-E®

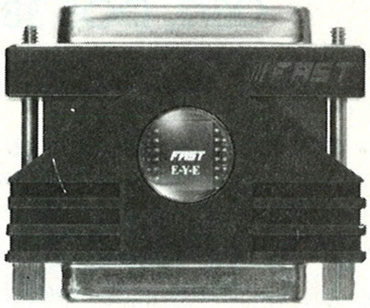


What hackers dislike . . .
Hardlock E-Y-E was designed using cryptographic principles. It took the experience and know-how of Germany's No. 1 in software protection and the leading edge technology of a US semiconductor company to create the ultimate software protection tool. Hardlock E-Y-E is based on a custom chip featuring secure algorithmic response rather than simple bit swapping or counting schemes.

What software developers like . . .
Hardlock E-Y-E combines all the features software developers require in a single product: algorithmic response to provide security and an optional non-volatile memory to allow custom configurations. FAST Electronic has made implementation of Hardlock E-Y-E in your software easy. Use HL-Crypt to protect .EXE or .COM files, or incorporate high level language interface routines in your software. The algorithm parameters and the contents of the memory can be programmed in seconds using our Crypto-Programmer card. This unique card guarantees that no one else can burn your original codes. Simply plug the card into any PC slot and start up your own Hardlock E-Y-E workshop.

What your customers will like . . .
Hardlock E-Y-E allows unlimited backup copies. The device is shipped with the software for the user simply to plug into the parallel interface and forget. Daisy chainability, outstanding reliability (no battery is needed), and the most compact High-Tech design ensure that your customer will accept Hardlock E-Y-E.

What your accountant will like . . .
Hardlock E-Y-E needs no factory coding. This ensures optimum delivery schedules and stock flexibility. Revenues will go up as software piracy and multiple usage are prevented. Despite its wealth of features, Hardlock E-Y-E's prices remain competitive.



Hardlock E-Y-E
programmable, algorithmic response and memory option – all in one.

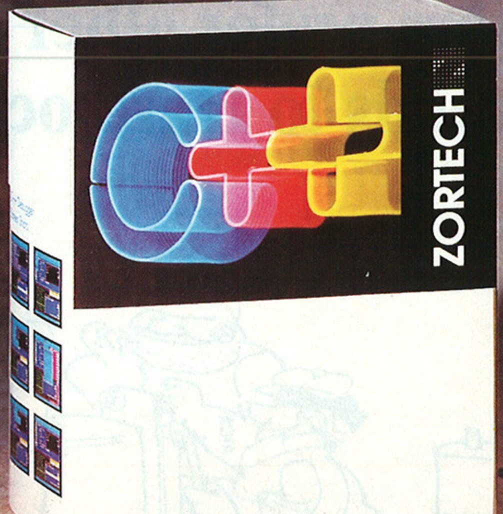
. . . As more and more software developers, customers and accountants appreciate the Hardlock E-Y-E device, hackers like it less and less.

MAGNIFEYE

Order your demo unit today. Contact Magnifeye,
235-239 Walmer Road, Walmer Studio #6, W11 4EY, Telephone 071 221 8024, Fax 071 792 3449.

CIRCLE NO. 571

C++ DEVELOPER'S EDITION V2.1

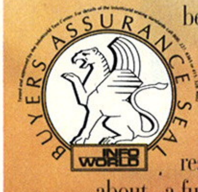


For total portability right now, your choices are a bit limited.

Only Zortech Gives You Total Portability to MS-DOS, Windows, OS/2, DOS 386, UNIX 386. And Macintosh. Right Now.

What platform will your applications be running on this time next year?

Now that C++ has made significant inroads into mainstream application development, you must choose the C++ compiler that is going to be the performance and portability leader now...and years to come. Since 1986, we've been testing, refining and perfecting our family of C++ products tirelessly. The results are what you are reading about, a fully portable family of C and C++ products available right now.



We Intend To Keep Up This Pace.

In the past few years, Zortech has quietly become an industry standard earning acclaim from Byte, Computer Language, Dr. Dobbs, and many industry experts for our solid implementation, market-leading performance, and strong, relentless innovation.

Whatever Platform You Choose, Choose Zortech Performance.

With Zortech, no matter what platform you choose to develop on, a substantial performance increase will be realized due to the quality of the original Zortech C++ implementation. For instance, on MS-DOS, your program will compile up to 35% faster with the resultant code running up to 45% faster and 25% smaller than Turbo C++*. But we don't stop there. On each platform, Zortech C++ is designed to improve your productivity where it really matters... in the development cycle.

For example, using the DOS/MS Windows package, you can edit, compile, link and debug the largest MS windows applications...and never once leave the Windows environment. All this plus complete plug-and-go to CNS C++/View and Commonview II! And, of course, these advantages are also available on the OS/2, DOS 386, UNIX, and Macintosh platforms.

60 DAY MONEY-BACK GUARANTEE

Choices. Choices. Choices...

If there is one constant in C++ development, it's change. The Zortech family of products keeps your options exactly how they should be. Open.

To find out more, call today.

Order Hotline:
44-81-316-7777

Zortech

Industry Standard. Industrial Strength.

Zortech, Inc., 4-C Gill Street, Woburn, MA 01801 Tel: (617) 937-0696 Fax: (617) 937-0793
Zortech, Ltd., 58-60 Beredford Street, London SE18 6BG Tel: +44-81-316-7777 Fax: +44-81-316-4138
Supported systems include Compaq, ISA/EISA, IBM PS/2 and 100% compatibles

* Source: MS-DOS, C++ Compilers, A comparison by the Ladd Group, (c) 1990

Taken as a whole, the product duplicates the standards of the SDK utilities, and in most areas considerably advances them. Larger resources of all kinds are now editable, dialogue designs are easier to manipulate, and the design of menus and accelerator keys in particular benefit from a GUI approach. The Microsoft utilities win on only two counts: Whitewater has omitted to include the SDK's in-program dialog test facility (incomprehensibly, given that it makes such a deal of a similar feature in its menu editor). And it also neglects to provide developers with a font editor: perhaps no bad thing, given the influx of professional font designers into the Windows world.

Turbo Debugger

We've given short reviews of Turbo Debugger in the past, largely commendatory. It's always been a well respected DOS source debugger. As a Windows debugger, it really hasn't any competition at all. The only currently available equivalent is the CodeView version included in the SDK, and that is slow and requires a second monitor. Turbo Debugger, on the other hand, operates much as it does in DOS, paging the Windows display out, and its text-mode interface in, whenever a breakpoint, trapped Windows Message, or Ctrl-Alt-SysReq keypress is detected. All the standard facilities from TD for DOS are there, including back-tracing, class hierarchy displays, and variable inspection. Additionally, two extra windows can display (in a readable form) global and local heap data and a log of decoded Windows Messages. Otherwise complicated Windows features are dropped seamlessly into the Debugger's operation. Take DLLs: if a traced program calls one, the libraries' source and symbols are automatically loaded up. Current DLLs can be listed and new ones patched in.

This package will sell just as much on the bundled utilities as it will on the C++ compiler

Again, Turbo Debugger manages to integrate and improve upon the functionality of three SDK programs - Spy, CV, and Heapwalker. It isn't perfect - whatever it does to wrestle control from Windows, for example, does a very good job of trashing my VGA card's palette. And I can't imagine Turbo Debugger remaining the only decent Windows debugger for long - expect swift retribution by both Microsoft and Multiscope, for example. But for the time being, this really is a considerable improvement.

Anything else?

I've barely enough room to cover the other changes to the C++ package. The assembler and linker have both been extended to allow segmented-executable compilation, and full high-level language PROCs. The Profiler, sadly, has been rather passed by all this Windows activity, and remains firmly DOS based. I'll just say that it works well, but is nothing to write home about.

One definite improvement upon Turbo C++ is the documentation, at last, of the small C++ class library provided on disk. It contains a generic Object class, with some common containers and lists derived from it. Sadly, the only flash of a Windows class library is provided in WHELLO.CPP (you can just catch a fragment of it in Figure 3), and this is of rather limited use, using a technique for storing the `this` pointer which is unusable with Windows' own predefined classes. Given the expectation held by many that Borland would be implementing a Turbo Vision-esque library for Windows, they did rather blow it here. This is clearly going to be a popular package, and while it's all very well organising OOP World Tours and so on, if you don't standardise on a decent object library now, and then get people using it, no-one will be able to inherit anything from anyone else. A decent class library might have delayed the launch of Borland C++, but might also have stopped people just using it as a decent C for Windows compiler, and let them start to take real advantage of its OOP features. There: sermon over.

A Foregone Conclusion

Shrewd readers may have already noticed a reviewer struggling to find anything uncomplimentary to say about this package, and it is at this point that I have to admit defeat. Borland C++ really is a considerable step forward for Windows programmers. It is intuitive, responsive, and well thought out. The Windows extensions have been carefully considered, not just bolted on. Features like pre-compilation of headers and the bundling of so many utilities make it still competitively priced in the DOS field, let alone in the barren Windows arena. That said, the other DOS compiler manufacturers are not stationary targets: Zortech is preparing a competitively priced rejoinder, JPI has its C++ within weeks, and Microsoft's C++ compiler is waiting in the shadows. Borland C++ is not quite the SDK - there is no help file compiler, for example, and the missing font editor could irritate. And Borland's licensing arrangement with Microsoft means the latter still holds the initiative. Borland C++ is just the first shot in the newly opened Windows range: nevertheless, it is a very well aimed shot indeed.

EXE

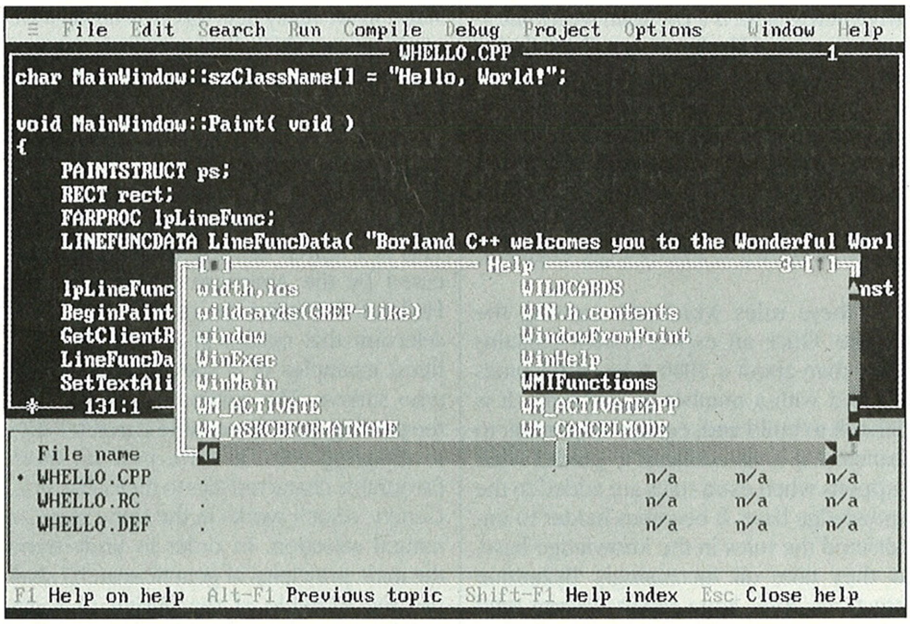


Figure 3 - A Windows, Object-Oriented 'Hello, World'

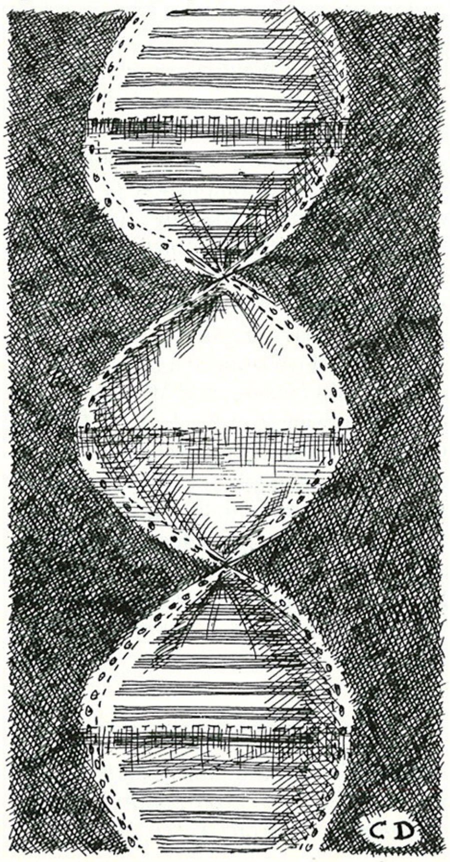
ADVANCED BUSINESS GRAPHICS

Borland C++ was announced on the 12th of February. It costs £300.00. Borland's product information line is on (0628) 771070.

Answer in the genes?

Expert systems have hit hard times. Darrel Ince surveys some techniques being used to dig AI out of the rule-based mire, including the fashionable genetic algorithm.

Searching forms a central part of all software development. Search methods are found in primitive forms in virtually every software system that has been developed, for example, as subroutines to search tables. They are also found in very much more advanced forms in artificial intelligence research. Search has its most public manifestation in the expert system: a program which attempts to replicate the problem-solving capabilities of the human consultant. Such programs search for some solution to a problem guided by rules which have been extracted from a human consultant. While we have sorted out most of the search problems in conventional software development, many remain as open research questions in artificial intelligence. For example, one problem currently facing expert systems researchers is that once the size of an expert system reaches a certain point, a number of undesirable events occur. In order to explain what these are, a short tutorial about expert systems is required.



During the 1980s, AI researchers were prepared to make increasingly strident claims for their technology

An expert system is a program which interrogates a stored file, known as a knowledge base, that contains the rules which are used by a human consultant in order to carry out some task which requires high-level thinking skills. These rules are extracted from the human consultant by the artificial intelligence equivalent of the systems analyst: someone known as a knowledge engineer.

It is these rules which are causing the trouble. Once an expert system contains more than about a 1000 rules, it becomes afflicted with a number of problems. It is difficult to build and, equally important, to maintain. It is also difficult to predict what happens when extra rules are added to the knowledge base. It becomes harder to understand the rules in the knowledge base, as they take on increasingly Byzantine structures. Also, large expert systems tend to become so grossly inefficient as rules are added that, after a certain point, consultation becomes impractical in real time. And it becomes very arduous to tease out the rules in the knowledge base from an expert. Even for small expert systems, with a few hundred rules, this is quite problematic; for knowledge bases with thousands of rules it is impossible.

Most expert systems carry out a search process. For example, an intelligent doctor's assistant searches through possible diseases guided by the rules in a knowledge base; an intelligent tool for detecting malfunctions in a machine tool searches for possible reasons for a malfunction, and a pattern recognition program that recognises defects in electronic circuits searches for defect patterns in a database of possible errors.

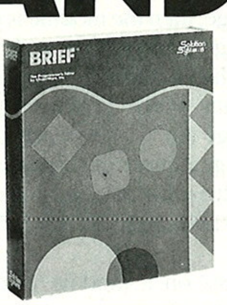
It is not surprising, then, that researchers have been attempting to look at solutions to the problem of search, where the space of possible solutions is very large and where the criteria for locating a successful solution are fuzzy. It is this area of software engineering which has spawned some of the weirdest techniques in computer science.

Weird Genes

Probably the weirdest is genetic search. This is a search method which was pioneered by the American researcher John Holland. It mimics the process of natural selection that occurs in nature, with the fittest examples of a species being those who survive. The survival is achieved by the passing on of favourable characteristics to offspring who, in turn, pass on more favourable characteristics to their offspring. Genetic search works in the same ways as natural selection. In order to understand the main principles of genetic search I shall use the example of the design of a gas pipeline, the type of application which is amenable to this form of search approach.

U.K. VERSION
NOW SUPPORTS BOTH
DOS & OS/2, DUAL MEDIA

POWER AND FLEXIBILITY



BRIEF 3.0 increases your productivity for just £199.00*

*VAT extra, FREE delivery

BRIEF has established itself as the leading programme editor on the market today. Now, with the launch of the latest enhanced version – BRIEF 3.0 – it offers even better productivity.

ENORMOUS UNDO POWER

BRIEF 3.0's enhanced UNDO capability takes the fear out of programming. BRIEF 3.0 allows you to undo any command that affects the cursor or the text (including global changes) up to 300 times, even back to the beginning of your session. It's the safety net you need to program with confidence.

REAL FLEXIBILITY

There's built-in support for 36 languages including Assembler, BASIC, C, COBOL, dBase, FORTRAN and Pascal. And thanks to its complete macro language you can make BRIEF 3.0 fit your own programming style with ease: change key assignment, commands, prompts, colours – even the indenting styles.

BRIEF 3.0 – new benefits

- With our new C-like macro language, CBRIEF, C programmers can write macros on the fly and not have to switch between C and a different macro language
- The new source level debugger works with both the original BRIEF syntax and the new CBRIEF macro language
- Maximum lines per file has been increased to 4 billion
- We've added to and improved template editing and smart indenting for ADA, Cobol, Basic, Fortran, Modula-2, Pascal and C
- We've added editable, multiple keystroke macros that can be saved and restored
- You can even save and restore your window set-ups

Plus the features that made BRIEF famous

- Flexible windowing
- Unlimited number of files
- File size limited only by desk space
- Regular expression search and replace

No other editor puts this much power and flexibility in your hands. So if you're looking for extra productivity purchase your copy of BRIEF 3.0 today.

Solution Systems

SOLUTION SYSTEMS are the UK distributors for BRIEF 3.0. When you buy from us you get the authorised UK version of BRIEF 3.0 dual media (3½" and 5¼") plus bonus macro diskette.

Special phone line for Access/Visa card holders. Save time by phoning 0763 244141 between 9.00 and 6.00 Monday to Friday to place your order.

N.B Official purchase orders from local authorities, PLCs and educational establishments are accepted by post or by Fax.
Fax no: 0763 244025

BRIEF 3.0 The Programmer's Editor

Send to: Solution Systems UK, The Maltings, Green Drift, Royston, Herts SG8 5DB

Please rush me my copy of BRIEF 3.0

☐ I enclose a cheque to the value of £228.85 (£199 + V.A.T.)

Orders despatched same day.

NAME ______________________ ADDRESS ______________________

______________________ POSTCODE ______________________

Please charge my Access/Visa* account by the sum of £228.85 (£199 + V.A.T.)

Card No: ______________________ Expiry Date: ______________________ Signature: ______________________

* Delete as appropriate

CIRCLE NO. 573

The problem is to design a gas pipeline so that costs of the pipeline are minimised: the cost of raw materials, the cost of paying landowners for permission to pass over their land, the cost of the pumping stations involved and so on. The first step in a genetic algorithm is to generate randomly a number of first attempts at the solution which satisfy the aim of generating the required amount of gas between two points. A number of solutions are then selected from this set, based on some objective criteria of the goodness of the solution. In the gas line problem, this may be the overall cost of the labour used to build the pipeline. This is the part equivalent to the natural selection process that governs the success of animals and plants - sickly individuals are culled without getting the chance to breed. After this, the smutty bit: the designs are mated with each other to produce brand new designs. For example, a new pipeline design may be constructed which contains sections from both its parents.

Finally, the solutions suffer mutation. This involves random changes to the designs which were formed by mating. There is a lot of confusion about the role of mutation in both genetic algorithms and human genetics. The reason why mutation is employed is because, if just mating and natural selection was used, then there is a potential for some good parts of a solution to be lost.

When a new generation of designs has been formed by natural selection, mating and mutation, the process continues with the development of a further generation using these operations, until a solution is found which matches the criteria for a good solution. In the gas-line problem, this would probably be some overall cost figure for the implementation of the pipeline.

Genetic algorithms have had quite a chequered history. In the early 1980s they were regarded as something of a backwater. Researches concentrated on the rule-based strategies that were used in expert systems. Now that expert system technology is beginning to falter, there has been a massive explosion in interest. Over the last five years, researchers have applied the techniques to a number of different problems.

Applications

One of the most impressive applications of genetic algorithms occurred in image registration in an area called digital subtraction aniography. A typical example involves a doctor examining a defective artery by injecting some dye into the artery, and then examining two x-ray pictures: one taken before the insertion of the dye, another taken after the insertion of the dye. The two x-rays are digitised and then subtracted, pixel by pixel, producing an image which outlines the interior of the defective artery.

Large expert systems become so grossly inefficient that consultation becomes impractical in real time

Unfortunately, slight movements of the patient during the period between the taking of the first and second x-ray pictures can spoil this subtraction process. Two American researchers, Fitzpatrick and Grefenstette, have used a genetic algorithm to discover the transformations required to bring the two x-rays into conjunction. Other applications include the design VLSI circuits, the configuration of computer keyboards, to find the optimal strength of landing struts on a fighter plane and to improve the design of a telecommunications network.

One area of software development which has been actively researched as an application for genetic search is the automatic generation of test data. At the late stages of a software project, programmers will be given the specifications of the modules that make up the system which they are to implement. A programmer would take the specification for a module, code the module and test it with sample data. This testing is a two stage process. First the programmer develops test data which checks out what the module does. So, for example, if the module sorted data in ascending order then a typical set of tests would involve one integer, the maximum number of integers, a collection of integers which is almost sorted and so on.

Once the functional tests have been carried out and the correctness of the module confirmed, the programmer then checks out the structural coverage of the tests that have been carried out. If the tests have achieved a high structural coverage, for example, 100% statement coverage and 85% branch coverage, then the module is passed to staff carrying out integration and system testing; if not, then the programmer develops more test data which increases the structural coverage above the standard specified by the programmer's quality assurance department. It is this structural testing which is tedious and can take a large amount of time.

The tests on a module involve selecting values for the parameters of the module and the global variables that the module uses, causing specific paths in the module to be executed. Researchers at the Open University are using genetic algorithms to generate structural test data. They are generating a random set of test data, monitoring the effectiveness of the test data by examining structural coverage, and then producing the next generation of test data by promoting and combining the best tests to form the next generation. This process continues until a test set has been produced which achieves the required structural coverage.

Pleasure and pain

Genetic algorithms look highly promising, especially where there is a large amount of non-linearity in the search space. Other promising techniques use the emerging technology of machine learning. This area of research has a pedigree almost as long as the history of computing. The pioneer British computer scientist, Alan Turing, in a paper written in the late 1940s, proposed

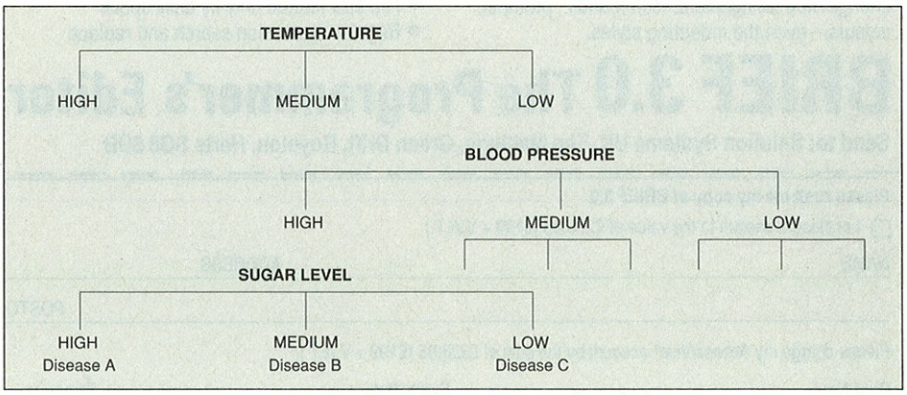


Figure 1 - A decision tree for medical diagnosis

that computer programs could be trained by means of what he called 'the pleasure and pain principle'. He proposed that computer programs were very like young children who increase their level of skills when they are praised, and who give up undesirable behaviour when they are punished.

Although researchers regard Turing's ideas as a somewhat crude description of current computer learning technology, they readily accept him as their intellectual progenitor. Current computer learning technology involves programs based on so-called induction algorithms. These algorithms work in the following way. An induction program is represented with a number of cases involving a series of factors, together with an outcome from each factor.

For example, the induction program might examine a series of diagnoses using factors such as patient temperature, blood pressure and the results of pathological tests. The outcome in this example would be a diagnosis of a particular collection of diseases. The induction program examines each case, and attempts to relate the outcome of each case to the variables. For example, it might find that a particular range of blood pressures and some tests always give a particular disease. This information is incorporated into a data structure known as a decision tree. The decision tree contains a codification of the reasoning processes which underlie the outcomes. Part of the tree for the diagnosis example is shown in Figure 1. Each path through the tree contains the combination of factors which correspond to a particular disease. For example, there is a path through the tree which says if a patient has a low temperature, high blood pressure and a high sugar level then the patient is suffering from disease A. Normally the decision tree will be much more complicated than this and would have a large number of levels.

Once the decision tree has been constructed from the test cases, a simple program accesses this tree, and then provides the advice for a doctor who wishes to use it. The main advantage of computer learning technology is that it eliminates the need for the painful process of eliciting rules from a human consultant. One disadvantage is that it can be very difficult to understand why the reasoning processes identified by the induction algorithm seem to work. Another is that when a problem changes, for example, a new factor is introduced, the induction program has to be run again. Probably the most impressive example of the use of computer learning has occurred in Australia. Researchers at the University of Sydney have used an induction program to reconstruct the reasoning processes which were used by bank staff to judge whether a customer is to be given a credit card. An induction algorithm due to the Australian computer scientist Ross Quinlan was used to develop an expert guidance system for staff at the bank.

There is a lot of confusion about the role of mutation in both genetic algorithms and human genetics

Neural net

Another approach to computer learning, which is closest to Turing's pleasure and pain principle, is known as *connectionism*. This uses a device known as a neural net to recognise patterns. A neural net is either a software or hardware structure which is taught to recognise the patterns that are presented to it. Researchers in this area often claim that their techniques mirror the processes which occur in the brain when patterns are recognised: that in the same way that the neurone structures in the brain configure themselves to recognise scenes, faces and structures, the software and hardware structures that are used to implement neural nets are changed by altering linkages and configurations by informing them of a correct response or an incorrect response.

One of the classical applications of neural networks is of face recognition in a security system. A video camera monitors a visitor at the door, opening the door if the individual is recognised as an employee. As preparation for the use of the neural system, each employee would be asked to present his or her face in a number of ways: smiling, grimacing, frowning and with a bland face. After all the employees had been presented once, new versions of existing faces would be presented, and the system trained by a human operator communicating correctness when a face was recognised, and failure when a face was not recognised.

Neural nets are now big business in America: special purpose hardware (such as the connection machine) has been developed to represent the hardware versions of the software structures that were used in the early research on neural nets; start-up companies have proliferated in New England and California; and there is a large amount of money available for research. Applications for which neural nets have been used, or are being researched, include: the recognition of bank notes, the recognition of gas chromatograph patterns when identifying unknown chemical compounds; and predicting the course of the Stock Exchange.

The future of AI

The whole area of artificial intelligence is now at a crossroads. Many commentators are increasingly recognising that, at best, the expert system has been a very limited success: while many undergraduates now construct expert systems with a small number of rules, the goal of expert systems with thousands for rules still eludes researchers; there has been a very high failure rate in artificial intelligence companies; workers in the area report to me that, although they still attract capital for feasibility projects, they cannot get the funds for real projects of equivalent size to those in non-artificial intelligence development. Much of the Japanese activity in this area has been reduced in favour of neural nets; and research funding in the area, particularly from the EEC, is a fraction of what it once was.

While the area of artificial intelligence is a fascinating technical one, it is much more interesting sociologically. The explosion of research in artificial intelligence over the last 10 years is a testament to the power of the Japanese: it was fear of their fifth-generation computer project which started the whole expert system bandwagon going. It is also a testament to the desperation of artificial intelligence researchers who, during the 1960s - 1970s had been starved of funds, and were, during the 1980s, prepared to make increasingly strident claims for their technology. What the artificial intelligence community now has to ask itself is this very difficult question: should we continue with a new generation of expert systems, should we shift our money into new technologies such as neural nets and genetic search, or attack on a broad front spreading the available funds into a diverse selection of technologies. This is an immensely tough question, one which I am happy that I do not have to answer.

EXE

Darrel Ince is a Professor of Computing Science at the Open University.

NOVELL Develops in DATAFLEX

The Benefits....

Buyers say...

"If I develop in DOS, I don't want to have to rewrite in OS/2" said Jim Baird, MIS manager for Novell Inc., in Provo, Utah. "DataFlex provides a good migratory path to OS/2 without losing what I have already developed in the DOS environment.

"With DataFlex under OS/2," he added, "I can put in one workstation and have a user do what would normally take two machines under DOS."

With more than 140 users accessing as many as 200,000 records and having to maintain a response time of less than one second, Baird likes the product's performance.

"I haven't had any problems with the corruption of data," he said. "I have users running 10 different OS/2 windows who can be working in a spreadsheet and come back to find the reports."

Baird likes DataFlex OS/2 as a development environment because of its macro capability for quick and simple programs and its solid programming language for more experienced users.

"You could start writing in lower-level code if you wanted to develop a quick program, and you could have a prototype out in 10 or 15 minutes," he said.

P.C. WEEK (U.S.A.)
February 19th, 1990

DataFlex provides quick applications prototyping and development through its relational database features and solid 4GL programming language. Once developed, this application can be easily run on DOS Networks and OS/2, as well as AIX, Xenix, Unix andVAX/VMS. Thus protecting the user's investment.

Novell are not alone in their use of DataFlex for applications development. Central and Local Government departments, corporates and software developers are just some of those who have been choosing DataFlex since 1981 - creating a library of thousands of successful DataFlex applications.

DataFlex is a powerful applications environment geared to on-line transaction processing and providing rapid access to data for large numbers of users. DataFlex not only protects your application development through technology portability, but also through technology developments, with the new options of a Database Engine, distributed database, Object Oriented Programming, CUA interface and relational report writer.

Try it out for yourselves with our special introductory offer of an evaluation copy with full documentation at £99 (including VAT and carriage). Send back the coupon below or phone our sales desk on **071-729 4460**.

The performance database for applications development

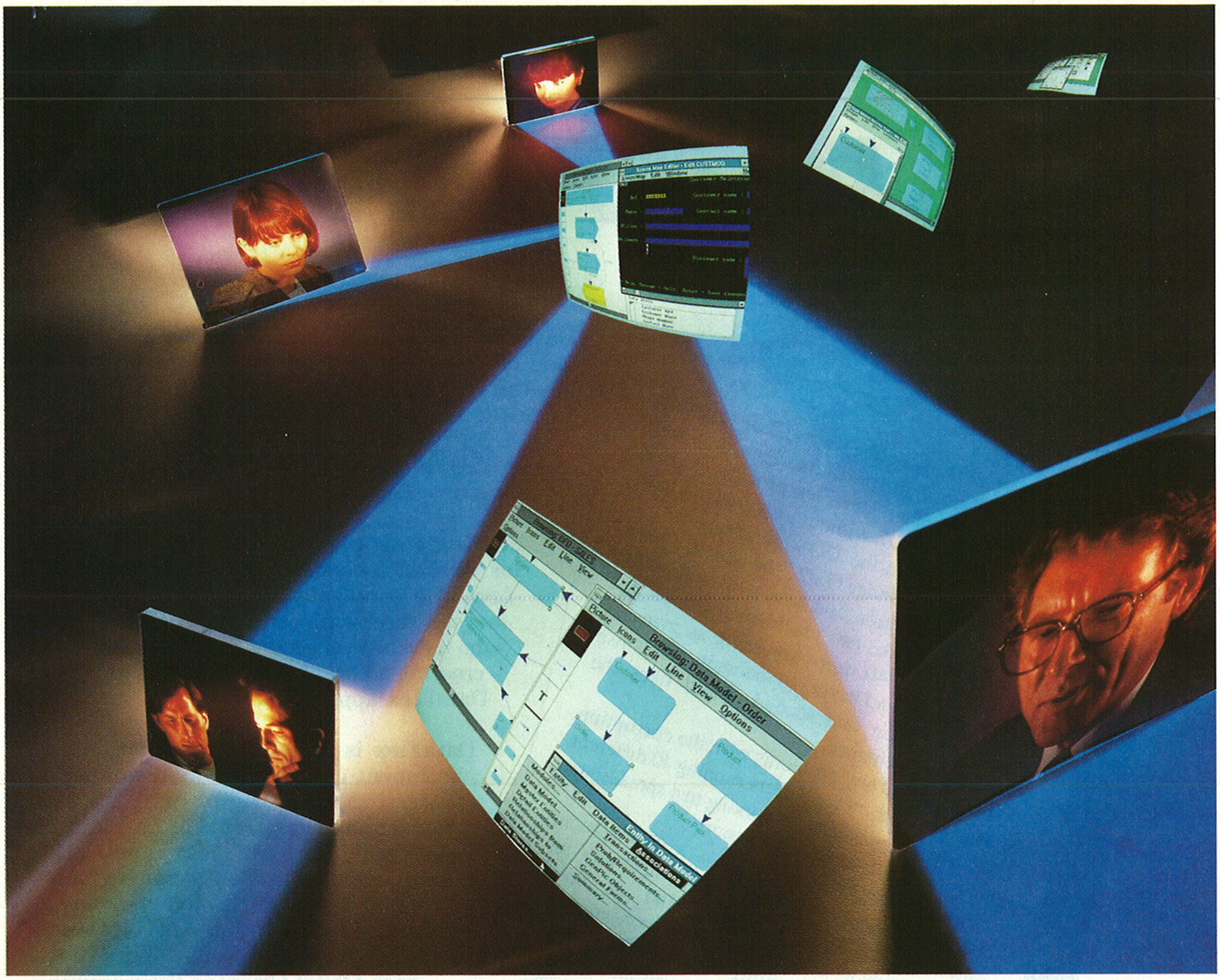
Further information from your **DataFlex** *dealer or:*

DataFlex (Information Management) Services Limited
DataNet House, 114/116 Curtain Road, London EC2A 3AH
Telephone: 071-729 4460
Telex: 94018587
Fax: 071-739 1247

All Trademarks acknowledged

CIRCLE NO. 575

Please send me information about **DataFlex.** EV
Name ______
Company ______
Position in Company ______
Address ______
______
______ Post Code ______
Phone ______ Date ______
What applications areas interest you? ______
______
DataFlex (Information Management) Services
114/116 Curtain Road London EC2A 3AH EXE 3/91



Software from hardware

Microway's range of 32-bit DOS compilers has been around for a while. Will Watts decided it was time to take a look at the C offering.

Here is a subject which is overdue for treatment in .EXE Magazine. C has long since been established as the favoured professional language for MS-DOS. But most of the current popular compilers do not take advantage of the recent growth in hardware. Many of us have 32-bit machines sitting on our desk, perhaps with powerful coprocessors and heaps of extended memory - but it isn't getting used. We are still writing 16-bit programs, struggling to fit them within the conventional DOS environment. It's all very well to use a driver to convert extended memory into a bit of disk-caching LIM, or perhaps use a multi-tasking OS to run the 386 in virtual 8086 mode, but this clearly isn't using the machine *properly*. This is particularly disappointing in view of the fact that as long ago as July 1987, .EXE ran an article explaining and comparing the benefits of three 80386 C systems (Intel C-386, Green Hills C386 and Metaware High C/Phar Lap).

I say that the time for fudges and stop gaps is past. The sexy-C market is maturing rapidly, and there is now a wide and ever-growing selection of compilers and tools. This is intended to be the first in a series of articles taking a look at what is available. I will be looking at the current 32-bit MS-DOS systems, plus throwing in a few odd-ball surprises. The package that concerns this article is one of the less well-known in the field: Microway's NDP C-386 system.

Microway

Microway, for those of you who can't place the name, is a Massachusetts-based company (with a UK operation in Kingston). Probably you are most likely to have encountered its adverts in the pages of PCW or other more hardware-oriented journals; it produces a range of cards, coprocessors and other go-faster add-ons with wonderful macho names like 'Number Smasher'. The company's products include plug-in transputer and i860 systems, and fancy 80x87-compatible coprocessors which outflank the Intel originals by performing more instructions per clock. This hardware background has influenced the design of the software, as we shall see.

The compiler dealt with my test application like a dog snapping a wasp

The NDP C-386 V3.1 compiler is part of a large family. The company also offers C++, FORTRAN and Pascal compilers; within the C range itself there are separate 486, 386 and 386SX compilers. The 386SX is a cheaper, semi-crippled version of the 386 compiler: it runs only on 386SX-based machines, and does not offer Weitech coprocessor support. The 486 compiler, on the other hand, can generate object code that is aligned to make the best use of the 80486 instruction pipeline. It even contains an inline assembler which performs these same optimisations on hand-coded assembly language, although, if the manual is to be believed, this is in a rather raw state in the current release: '...the assembler does not flag illegal syntax or addressing, and the output in such situations is not defined... all this may sound scary.' They're not kidding. Code which has been optimised for the 486, by the way, may still be run on a 386/387 machine; but it will run slower than 'ordinary' 386 code.

To return to the C-386; the compiler is available for UNIX, XENIX and MS-DOS. The latter case obviously requires some sort of DOS extender to make it work. Earlier versions required you to purchase the Phar Lap toolset; the current offering, while retaining compatibility with the previous system, is supplied with a set of NDP tools based on the Ergo DOS Extender. I have concentrated on the NDP system, since Phar Lap support is standard in this class of software, but the NDP Tools are special.

Hardware and installation

The manual says that you need a 386SX/386DX/486 PC, with at least 2 MB of extended RAM (4 MB recommended) and 3 MB of hard disk. If you want to do floating point calcs, you will also need some kind of coprocessor - this can be the FPU of your 486, a 287, a 387(SX) or, if you are really going for it, a Weitech 3167/4167. For the record, I have been using a 20 MHz 386 machine equipped with 3 MB extended memory and a Cyrix 387-compatible coprocessor.

The software is contained on five 1.2 MB diskettes: three for the compiler and libraries (with some duplication of function in order to support both Phar Lap and NDP Tools), one containing the NDP Tools (linker, librarian and the DOS extender itself), and one containing source and executable for MIKE, a Micro-EMACS like editor. I don't speak EMACS, and when I couldn't find its commands documented in the manual I'm afraid that I decided to take this software on trust. Since most programmers will not switch editors for any lesser reason than a really good integrated environment, I assert that this doesn't matter very much. You get two versions of MIKE: a 16-bit and a 32-bit protected mode, so multi-megabyte source file maniacs should be content.

Installation is achieved via a series of low-tech INSTALL.BATs which use the public domain LHARC utility to expand compressed files. The batch files set up a slightly non-

standard (and, spit, mis-/un-documented) directory structure below the base \NDP30. The .H headers are placed in \NDP30\INC\ANSI and \NDP30\INC\INCLUDE, with many old stand-bys like STDIO.H duplicated across both. I *think* that the latter are a more K&R-oriented set, but I have not yet found this written down anywhere. The rest is pretty straightforward: executables live in \NDP30, libraries in NDP30\LIB; you need to set up a few environment variables (LIBP, INCLUDE etc) to tell the tools where to look for their component parts.

The compiler

Users of older versions of NDP C will know that, to get a piece of running code, you used to have to pass the output through Phar Lap's assembler to generate the object files. This piece of tiresomeness, which is more reminiscent of CP/M Absurdly Tiny C than an alleged heavyweight like NDP, has now been cured; the compiler creates its own .OBJs, although it can still produce assembly language at the drop of an option switch.

The compilation process is controlled by the MCC command, equivalent to Microsoft's CL, which calls the compiler and linker in turn to produce the final executables. These tool themselves require a fully equipped 386 machine to run (although they don't need the coprocessor). The output of a successful run is a .LTL file, which is equivalent to, but incompatible with, Phar Lap's .EXP files. To run the your application, you type 'NDPRUN <myapp>'. Unlike its Phar Lap equivalent, NDPRUN may be freely distributed with your applications - there is no licence fee.

There isn't a conventional MAKE tool in the package, so to produce multi-source file programs I used Borland's. This worked fine - MCC puts up no struggle when spawned by another process. Also, for the

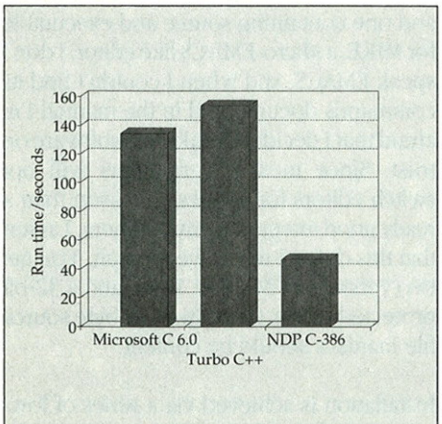


Figure 1 - Comparative run times for Analyse

purpose of the easy removal of compilation errors, I wrote a filter program to allow MCC to be called from within the Turbo C++ IDE. This is a TC++ feature which allows you to capture the screen output from a third party compiler and use it to drive the IDE's browser, and skip through source files from error to error at the touch of a key. Generally this worked, although there were occasional crashes which might have been caused by MCC, the IDE or my even improvised filter (all right, I can see which you blame). I think that the prospects of successfully hooking MCC into other similar products, such as Microsoft's Workbench, are pretty good.

A test application

The premise of this article is that you have a DOS-based application which is getting cramped (although this package could equally be used the other way round: porting down a VAX program to the PC). In order to present the software with a reasonable challenge, I have borrowed some real, commercial code to test. Analyse is a 2D plane frame analysis program from the Poole-based software house CADS. Analyse is designed for civil engineers; the input data describes a 2D section of a building, on which are placed various loads representing the weight of the construction materials themselves, the contents of the building, the effect of wind and snow and so on. The program uses the technique Gaussian elimination on a large matrix (its size is proportional to the square of the number of joints in the design) to calculate the way the building deforms under these loads. It also calculates various engineering mysteries such as shear and axial forces on the components.

For our purposes, it suffices to say that Analyse sucks in vast amounts of floating-point data, chews it all up, then blasts the results back to disk. Under DOS, it suffers from exactly the sort of difficulty that I have described. It quickly runs out of memory as the complexity of the design increases (because, in order to obtain a practical speed, the joint matrix must be kept in RAM), and it can take serious amounts of time to do its calculations. Ripe for porting, in fact.

Analyse is written in Microsoft C. It has a fancy user-interface involving much mice and icons, which for the purposes of this article I am leaving well alone (there being a limit to how much time I can spend programming before blank pages start appearing in .EXE). My tests were based on the 10 main modules which do most of the real work. I stubbed off all calls to do with the interface, but this left me with a problem: what to use for test input data. I couldn't use the files generated by the MS version directly, as Analyse stores its data in binary format by writing `structs` directly to disk. 32-bit compiler's `structs` would clearly be different sizes (because, for example, `ints` are held in 32-bits instead of 16) and in any case, C compilers are allowed to pad structures with unused bytes to force favourable alignments. This padding can differ between two 16-bit compilers, so I was obviously onto a non-starter as far as a 32-bit compiler was concerned. My solution was to write import/export programs, which converted the Analyse data to and from ASCII format. I compiled and ran the TOASCII program under Microsoft C, and the TOBINARY program with the test compiler, and in less time than it takes you to spend a week swearing at the vagaries of `scanf()`, I had a set of test data. Before embarking on the testing 'for real', I used this system to port Analyse to Turbo C++; this worked fine, producing exactly the same calculation results.

Porting to NDP

I knew that porting to NDP C-386 would require changes to the source; I had predetermined to minimise these and maintain the code compilable by the Microsoft and Borland products. NDP predefines the macro `__NDPC__`, so it was easy to contain changes within `#ifdef ... #endif` preprocessor directives. In fact, I had to do very little. NDP runs in the small 32-bit memory model, ie just 4 GB of address space, so it didn't like the keywords `near` and `far`; but this was easily solved by defining macros of those names which expanded to nothing. The compiler doesn't take the `//` convention for comments, so I was forced to substitute `/*...*/`s throughout. NDP's slightly eccentric library forced me to rewrite a couple of I/O routines. The nearest that Microway gets to `kbhit()` is a strange thing called `inkey$()` which appears to have been filched from an old BASIC manual.

The task completed, I turned on all the go-faster option switches that I could find, converted a set of data representing a section through a car-park, wound up my stopwatch and got testing. The results are shown in Figure 1. You will see that NDP completely routed the opposition, finishing in just over a third of the time taken by the faster 16-bit program (Microsoft's). This result, you should note, was obtained from an application which performs quite heavy disk I/O - this should disadvantage the 32-bit application, which must switch back to real mode every time it calls DOS services. I think this is really rather impressive; I was not, you recall, running some benchmark

TOOLBOX WINTER '91 NOW AVAILABLE

ASSEMBLERS
- 386 ASM Link & Locate £550
- Microsoft MASM v.5.1 £89
- OPTASM £85
- Turbo Debugger & Tools £99

BASIC
- Mircosoft BASIC PDS 7.1 £270
- BASIC Interpreter £215
- Microsoft Quick Basic £60
- True Basic £65
- Power Basic £89

C
- Lattice C Compiler v.6.01 £170
- Microsoft C v.6.0 £250
- Watcom C v.8.0 £255
- C-terp £195
- C86 Plus £325
- Express C £65
- High C-286 £385
- High C-386 £585
- Instant C £310
- Instant C/16M £515
- NDP C £385
- Quick C £60
- Quick C & Assembler £125

C++
- Borland C++ £129
- Borland C++ Professional £195
- Guidelines C++ with book £275
- Intek C++ 80386 £320
- Zortech C++ Compiler v.2.1 £120
- Zortech C++ Dev. Edition £270

COBOL
- acucobol MS-DOS £900
- acucobol Net-Bios £1250
- acucobol XENIX-286 £1500
- acucobol UNIX-386 £2000
- acucobol *OTHERS* CALL
- Microsoft COBOL v.4.0 *NEW* £585
- Real Menu £210
- RealCICS £750
- Realia COBOL £750
- RM/Cobol-85 £750

dBASE
- Clipper Version 5.0 £325
- DB Fast/PLUS £255
- DB Fast/WINDOWS £255
- dBASE III Plus £395
- dBASE IV £515
- dBMAN V £190
- dBXL Diamond £165
- dBXL/Lan £389
- Foxbase+ v.2.10 £255
- Foxbase+/386 £385
- Foxbase+/LAN £385
- Foxbase+ Runtime £445
- Foxbase+ SCO XENIX £495
- Quicksilver Diamond £389

DATABASE MANAGEMENT
- Advanced Revelation £615
- AskSam £190
- Clarion Personal Developer £195
- Clarion Professional £545
- Informix 4GL £645
- Informix 4GL for Xenic £900
- Opus One £255
- Oracle for 1-2-3 £195
- Paradox 3.5 £595
- PFS Professional File £165
- R:BASE Ext. Report Writer £190
- R:BASE for OS/2 £579
- R:BASE for DOS £445
- Reflex £89

FORTRAN
- Microsoft Fortran v.5.0 £250
- Fortran-F77L £385
- NDP FORTRAN-386 £385
- ProFortran £235
- RM/Fortran £385
- SVS Fortran £435

OTHER LANGUAGES
- ACTOR £450
- ACTOR Language Extension £65
- CCSM Mumps Single User £65
- PC Logo £65
- RPG II Complete £1035
- Smalltalk/V £65
- Smalltalk/V 286 £129
- Snobol 4+ £65
- Toolbook £299
- Trilogy £65

PASCAL
- Microsoft Pascal £170
- Oregon Pascal-2 £149
- ProPascal £235
- Metaware Professional Pascal £579
- Metaware Prof. Pascal 286 £385
- Turbo Pascal v.5.5 with objects £89
- Turbo Pascal Professional £159

PROLOG
- Arity Combination Pack £710
- Expert System Dev. pack £190
- Intelligence Compiler £299
- MicroPROLOG Professional £645
- MProlog £299
- Professional Prolog II Plus £899
- Professional Prolog £545
- Prolog Advanced Toolkit £99
- Prolog Compiler £129
- Prolog Compiler/Interpreter £420
- Prolog-86 Plus £165
- SQL Development Pack £190
- Turbo Prolog £65

AI AND OBJECT ORIENTED
- C-Talk £99
- C+ Objects Vol 1 with Source £275
- CxPERT £515

CASE TOOLS
- Easycase Plus £190
- Visible Analyst £385
- Visible Analyst Workbench £1535
- Visible Dictionary £385
- Visible Prototyper £385
- Visible Rules £385

COMMUNICATIONS LIBRARIES
- AdComm £190
- C ASYNCH MANAGER £125
- Communications Library £195
- Comm Pak with Breakout £195
- Multi Comm £89
- Net Lib £129
- NET-TOOLS £95
- SilverComm £125
- Comm Plus £65
- Turbo ASYNCH PLUS £125

FILE MANAGEMENT
- B-Tree £65
- B+Tree for C £55
- AXOS £255
- Btrieve OS/2 £370
- Btrieve Multitasking £370
- Btrieve for DOS v.5.1 £395
- Btrieve XENIX £370
- C Index+ £255
- c-tree £255
- Faircom Toolbox Professional £710
- Code Base 4 v.4.2 DOS & OS/2 £190
- Code Base 4 UNIX/XENIX £319
- db_FILE/RETRIEVE £190
- d-tree £320
- dBC III £165
- dBC III Plus £330
- RealDL/I £645
- WKS Library £129

GENERAL
- C Tools Plus £99
- C Utility £165
- C*Lib C Function Library £120
- DB/RA3 £125
- dBase Program Utility £55
- dBase Program Utility Vol 2 £55
- dBase Tools For C £55
- Functions with source £149
- Integrated Dev. Libraries £99
- Pascal Toolkit £85
- Pascal Tools 1 £75
- QuickPak Professional £99
- Quick Tools Library £79
- Spindrift Library £99
- Spontaneous Assembly £255
- Super Functions £195
- Tom Rettig Help Combo £79
- Tom Rettig Library £65
- Turbo C TOOLS £99
- Turbo Plus £129
- Turbo PowerTools Plus £99
- Turbo Functions £69

GRAPHICS
- dBase Tools for C Graphic £55
- dGE v.4.0 £190
- EGA Toolkit £165
- Essential Graphics £259
- Flipper £129
- FontWINDOW/Plus £170
- GEOGRAPH Professional £210
- Grafmatic CGA & EGA £135
- GraphiC £255
- GSS Graphics Dev. Toolkit £399
- GSS Kernel £385
- GSS Metafile Interpreter £225
- HALO-88 £255
- HOOPS Extended DOS £515
- MetaWINDOW/Plus £210
- scLASERplus £299
- SilverPaint £65
- SilverPak £179
- Slate £195
- Universal Graphics Library £130

OP. SYSTEM ENHANCEMENTS
- DESQview API C Lib/Ref £129
- Multi C £89
- Multi Dos Plus w/source £65
- OS/2 Programmer's Toolkit £215
- OS/286 £320
- OS/386 £320
- Resident C £65
- Time Slicer £190
- VMFM £115
- Wendin PC UNIX £60
- Wendin PC VMS £60
- Microsoft Windows v.3.0 £95
- Microsoft Windows S.D.K v.3.0 £325

QUERY
- CBTREE/SQL £255
- CQL Query System £265
- dQUERY/Lib £179
- db_RETRIEVE £239
- XQL £480
- Xtrieve PLUS £395
- Netware SQL £385

REPORT
- CB/IQ Report £139
- Coreports £179
- Focus Report Writer £210
- r-tree £179
- R&R Relational Reportwriter £95
- R&R Clipper Module £49

TEXT SCREENS
- Animator COBOL £120
- Application Manager £450
- C-Worthy v1.2 Forms/CW £320
- C-Worthy v1.2 Forms/CW+Src £450
- Curses £105
- Curses Window Dev. Kit £75
- Data & Windows £149
- DataWindow £179
- dWindows for db3 £60
- Face-it £65
- Flash-up w/Toolbox £85
- Forms II £179
- Guido £165
- HI-SCREEN XL £99
- Multi Windows £179
- On-Line Help £90
- PANEL PLUS II £319
- PC Forms £129
- Pop Screen £59
- POWER SCREEN £99
- Pro Screen £65
- SCREENIO £259
- Screen Sculptor £75
- ScreenStar £65
- Screenplay £115
- SoftCode £65
- SoftCode Template £39
- Softscreen/Help Source £150
- Sourcewriter £645
- UI Programmer Dev. Edition £345
- V Screen £39
- VCScreen £90
- Vitamin C £135
- VLib with Source £90
- Vermont Views £245
- Z View £149

DEBUGGERS & DISASSEMBLERS
- 386 Debug £125
- Breakout-II £79
- D'Scope 2 £159
- dBug for dBase £125
- DIS*DOC £79
- Periscope II £110
- Periscope II-X £90
- Periscope III £875
- Periscope I Rev 3 £435
- Periscope IV (16 MHz) £1245
- Periscope IV (20 MHz) £1435
- Periscope IV (25 MHz) £1625
- Plus Board with 512K £310
- Plus Board with 0K £220
- Serial Test £310
- Soft Probe II/TX £245
- Soft-X-plore £79
- Sourcer/BIOS Pre-Proc. *NEW* £110

DOCUMENTORS
- ASM Flow £69
- C-Clearly £95
- Clear+ for dBASE £129
- Clear+ for C £129
- dAction £55
- dAnalyst £190
- dFLOW £89
- Documenter £190
- Flow Charting III £149
- Tree Diagrammer £65

EDITORS
- BRIEF v.3.0 £159
- BRIEF v.3.0 *UPGRADE* £60
- BRIEF OS/2 £159
- dBRIEF v3.0 *NEW* £79
- BRIEF with dBRIEF £229
- C-EDIT £255
- EMACS £210
- Epsilon £129
- ME Editor with C Source £125
- Norton Editor £49
- Personal Rexx £99
- PI Editor £129
- SPF/PC £159
- Vedit+ £120

HELP GUIDES AND TRAINING
- dBASE on-line £65
- Norton Guides for ASM £85
- Norton Guides for BASIC £85
- Norton Guides for C £69
- Norton Guides for OS/2 £89
- Norton Guides for Pascal £85
- Programmer's Library £255

LINKERS & LIBRARIANS
- ALINK *NEW* £129
- Archive Link £239
- Link & Locate ++ £255
- LinkLoc £315
- PLINK86 Plus *Now Polytron* £495
- PolyLibrarian II £125
- .RTLink £190

MAKE UTILITIES
- PolyMake v.3.2 £205
- MKS Make £99
- MKS Make 5 User £320
- MKS Make 10 User £515

PROFILERS & OPTIMISERS
- Codesifter £79
- DOSTRACE £65
- PC Metric £129

PROTOTYPERS & GENERATORS
- Automated Programmer £320
- Cogen £290
- COMPEDITOR £130
- Dan Bricklin Demo II £129
- Data Boss for Pascal £395
- Genifer £255
- Instant Replay III £99
- Sycero db (U.K.) £595
- Sycero db Net (U.K.) £745

SYNTAX ANALYSERS & LINTS
- PC Lint v.4.0 £90
- Pre-C £150

TEXT & UNIX TOOLS
- C Programmers Comb. £75
- C ToolSet £65
- MKS LEX & YACC £165
- MKS Toolkit v.3.1 *NEW* £165
- MKS Toolkit v.3.1 *UPGRADE* £50
- MKS AWK £65
- PC Fortran Tools £99
- QPARSER Plus £195
- Spellcode £65
- Turbo Professional £85

TRANSLATORS
- BAS_C Commercial £320
- Bastoc £515
- dBx dBASE to C v.4.7 £389
- dBx Library Source CALL
- dBx UNIX/XENIX Source CALL
- FOR_C v.2.0 MS-DOS *NEW* £565
- FOR_C v.2.0 XENIX *NEW* £630
- FOR_C++ XENIX/386 *NEW* £1195

VERSION CONTROL
- M.K.S. R.C.S. £125
- M.K.S. R.C.S. 5 User £485
- M.K.S. R.C.S. 10 User £905
- PVCS Professional £395
- PVCS Network (5 User) £1395
- PVCS/PolyMake OS/2 CALL

BACKUP - RECOVERY - REPAIR
- dSalvage £65
- Fastback Plus £125
- Mace Utilities 1990 £95
- Norton Utilities Advanced £95
- PC Tools Deluxe £85
- XTree Net £255

COMMS. & TERM. EMULATORS
- Blast II £165
- Blast II w/Remote Control £199
- Carbon Copy Express £225
- Carbon Copy Plus £129
- Crosstalk Mark 4 £160
- Crosstalk XVI £130
- EM4105 £320
- Mirror III £65
- PC Anywhere III £95
- Sidetalk II £79
- SmarTerm 125 £190
- SmarTerm 220 £129
- SmarTerm 2392 £65
- SmarTerm 240 £225
- SmarTerm 320 £129
- SmarTerm 400 £115
- SmarTerm 4014 £145
- VTek £125
- Z Stem PC-VT100 £65
- Z Stem PC-VT220 £99
- Z Stem PC-VT240 £190
- Z Stem PC-VT4014 £65

DISK MANAGERS &OPTIMISERS
- HTest/HFormat £85
- Mace Gold £99
- Norton Utilities Advanced £95
- SpinRite II £59

OP. SYSTEM UTILITIES
- 386MAX 5.0 £135
- DESQview £85
- DESQview/386 £125
- Headroom £85
- Move'Em £59
- Norton Commander £99
- Omni View £55
- Qemm-386 £65
- Qemm-50/60 £39
- Soft Bytes 386 Max £45
- Weiner Shell £195
- Windows 3 £95
- X-Tree Pro Gold £85

FAX OR TELEPHONE CREDIT CARD ORDERS VISA OR ACCESS

PRICING
All prices exclude V.A.T., but include Overnight delivery.

EDUCATIONAL DISCOUNTS
10% Educational discounts available on all products.

CATALOGUE & FACT SHEETS
Call for our new Winter '91 '91 catalogue, over 165 pages of Software Development Tools.

THE SOFTWARE CONSTRUCTION CO. LTD.

1 THE MALTINGS, GREEN DRIFT, ROYSTON, HERTS. SG8 5DB (0763) 244114 FAX. (0763) 244025

CIRCLE NO. 577

to test for speed, but a real live application that is in regular use. Out of interest, I also timed the compilations themselves (Figure 2). In this field NDP is certainly no speed-merchant, but it came in well ahead of Microsoft C 6.0, which kept up a running commentary of warnings about all the functions it couldn't optimise, and all the debugging information it was having to throw away. The Microsoft compiler is *still* slightly slower than NDP with all the optimisations switched off, although it must be said, in fairness, that NDP does not provide the time-saving incremental compilation features included in the MS offering.

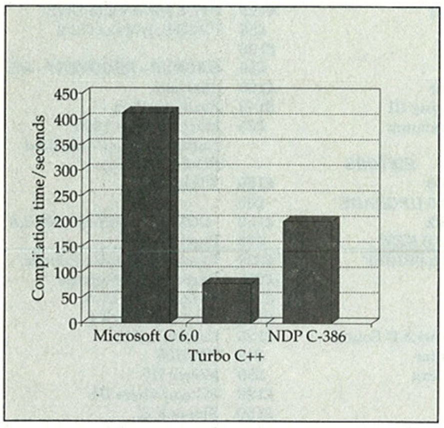


Figure 2 - Comparative compile times for Analyse

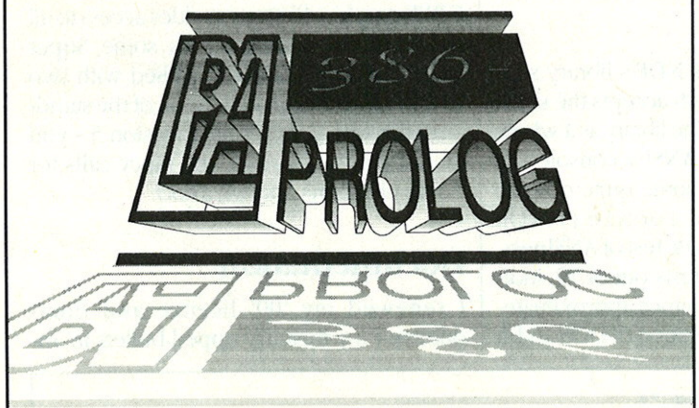
However, there was a problem. NDP produced different results to the 16-bit compilers. Not wildly different; in most cases much less than 1%, but sometimes rising to more significant differences, as in -1.8413e-016 metres (Microsoft/Turbo) versus -1.77046e-016 metres (NDP). Now I am fully aware of the limits of accuracy of floating-point arithmetic, nonetheless this was a surprise. All three compilers were supposedly operating to the same floating point accuracy (all Analyse quantities are declared as `doubles`, ie 64-bits), and were using the same coprocessor to do the calculations. Microway suggested that the difference might be caused by its compiler's more extensive use of the coprocessor. Internally, the 80837 stores numbers in 80 bits; this extra accuracy is lost as soon as a number is moved out into memory. NDP can assign variables and intermediate results within the coprocessor registers, retaining the extra 16-bits. However, when I recompiled the application with option switches that were supposed to force the compiler to keep all variables and temporaries in memory, the differences remained. Someone else suggested that an inaccuracy might be caused by problems with Microway's ASCII/binary conversion routines, but a simple test established that this was not so.

I can see two plausible explanations. Either there is some horribly subtle porting bug, hidden away somewhere in the depths of the calculations, or difference is caused by NDP and the 16-bit compilers making different choices about the order of evaluation of expressions. My bet is on the latter. Note that I have been careful not to call either set of results *wrong*. The downside of testing with a complicated program like Analyse is that it is hard to find out what the 'right' answer is; I now need to port the program to a machine which does 128-bit floating point maths to obtain a basis for comparison. A case of being well hoist with my own petard. I promise to let you know how other compilers fare in this comparison of accuracy.

Features and Libraries

NDP offers some interesting extra features. For those not possessed of Microway's Clearview source-level debugger (which included me for the purposes of this review), the compiler provides an option switch which

Think BIG!



Forget 640K. Forget EMS. Forget the 16-bit squeeze. With LPA 386-PROLOG, you get a genuine 32-bit programming environment, which runs directly under MS-DOS and yet is capable of addressing up to 4G (4096M) of memory. All you need is a 386 or 486, and the talent to Think BIG!

Logic Programming Associates Ltd
Studio 4, Royal Victoria Patriotic Building
Trinity Road, London, SW18 3SX, England
Tel: 081 871 2016 - Fax: 081 874 0449

CIRCLE NO. 579

MULTIPLE INTELLIGENT COM PORTS FOR OS/2

THE IMPAC 8 & 16 INTELLIGENT COMMS CARDS

* MICROSOFT COM X COMPATIBLE
* 8 TO 64 FULL MODEM PORTS
* REDUCE CPU OVERHEAD FROM 40% TO 1% PER PORT
* FITS ALL ISA (PC/XT/AT) MACHINES
* DOES NOT USE SHARED RAM
* PROVEN FIELD PERFORMANCE
* UK DESIGN AND MANUFACTURE
* DRIVERS FOR DOS, OS/2, UNIX & XENIX

DEALER ENQUIRIES WELCOME

For more information contact:
SCL Downswood, Carlton Road
Reigate, Surrey RH2 0JQ
Telephone: 0737-762200

CDOS DRIVER ANNOUNCED

CIRCLE NO. 580

QIC 'n Easy Access to all your Data Everytime

AS/400
Sys 36/38
Apollo
Everex
Sun
Sytos
UNIX

Lotus 123
DBASE
Mail Merge
Secure Data
Duplication
XENIX
UNIX

QICPAK

The QICPAK family provides unique facilities to access 1/4" (QIC) data cartridges created on a wide range of Micros, Minis and Mid-Range Systems using your PC. This gives you a secure, low cost, fast and high capacity alternative.

QICPAK's facilities cover all aspects of cartridge processing, including:

- Extracting data directly from the cartridge into packages such as Lotus 123 or DBASE or for use in Mail Merge applications.
- Data files can be extracted by QICPAK from cartridges recorded on many systems, including: IBM System 36/38, IBM AS/400, IBM PC-RT, Apollo, Everex, Maynard, Mountain, Sun, Sytos, UNIX and XENIX systems etc.
- QICPAK's High Speed Cartridge Duplication gives complete in-house control.
- QICPAK's Backup & Restore facilities are UNIX tar compatible
- Custom applications, eg recorded information may be protected from unwanted access providing secure interchange of your confidential information.
- Source in Microsoft & Turbo C, Turbo Pascal and BASIC is provided.

We provide both QICPAK Kits for use with existing 1/4" cartridge drives and also complete internal & external solutions. Support is by the developers.

Access & Visa Accepted

VOGON
VOGON ENTERPRISES LIMITED
94 Easthampstead Road, Wokingham,
Berkshire RG11 2JD
Tel: (0734) 784511/890042 Fax: (0734) 890040

CIRCLE NO. 581

forces the code to call a user defined routine `user_debug_()` before every function call. `user_debug_()` is passed the name of the calling routine, the line number of the call and the name of the function being called, so it is quite easy to set up a tracing system. There is also an option switch to perform run-time checking of array boundaries.

More glamorous features reflect Microway's extreme interest in giving you the ability to create demonically tight code. NDP offers an extension to C's `register` keyword that allows you to assign variables to specific registers. So this

```
reg$eax unsigned eax;
```

aliases the EAX register to the `eax` C variable. This is combined with an inline assembler `asm` statement, which is unusual in that it requires that you specify the registers read and written to by the assembly language fragment. This allows the compiler to make sure that its registers are in order before and after the `asm`. For example,

```
asm(ax,bx, " int 10h",cx,dx);
```

makes a call to interrupt 10h, warning the compiler that AX and BX are used as input, and that CX and DX get corrupted. Combining both these features, you can write a function like that shown in Figure 3, which Microway says compiles to more efficient code than most assembler programmers could produce.

For the less daring, the NDP's library supplies `int386()`, which accepts the usual `struct` register sets. The library as a whole is pretty good UNIX + ANSI + console I/O mixture, and includes some quite obscure ANSI calls, such as `setlocale()`. (On the Plum Hall rough-guide test of ANSIness, the compiler failed 17 tests out of 79, indicating a level of conformance approximately equal to the last generation of DOS 16-bit compilers, rather than to the likes of current offerings such as Microsoft C 6.0.) The GREX graphics library provides access to all the standard adapters plus some Super VGA systems. GREX is supplied with two APIs; its own and an emulation of the standard Microsoft C calls (circa version 5 - you do not get the more recent, fancy calls for doing pie charts and whatnot).

Documentation

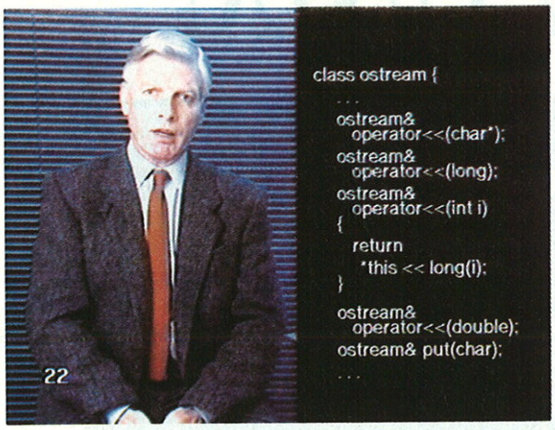
I renewed my '00' licence and dumdummed my mercury tipped bullets in an-

```
/* A useful Globals */
unsigned char page = 0;

void pc_test()
{
  reg$ah  unsigned char ah; /* declare the registers */
  reg$bh  unsigned char bh; /* ah, bh, dh and dl     */
  reg$dh  unsigned char dh;
  reg$edx unsigned char dl;

  ah = 3;
  bh = page;
  asm(ah,bh," int 10h", dh,dl);
  y = dh;
  x = dl;
}
```

Figure 3 - NDP's incline assembly syntax

VIDEO COURSE

Learn C++



NO RUSH!
NO TRAVEL!
NO HOTELS!
ONLY £299.95

Learn C++ Now!
The great exodus of programmers from C to C++ has begun! Since C++ builds on C, it's the easiest OOP language to learn. That's why it's called "the language of the 90's".

Why the rush?
PRODUCTIVITY! Yes, C++ programmers can write programs in less time requiring less maintenance. Large projects become much easier to manage.

Unfortunately, learning C++ can be very costly. Classroom instruction is expensive even without the travel and hotel costs. Of course, not learning will cost you even more in the long run. Now there is an alternative!

The top C++ video tutorial at the lowest possible price.
The C++ video tutorial from Zortech is the ultimate C++ training tool for work or home at only £299.95. It comes on six VHS video tapes containing 32 clear, extensive tutorials.

Used in conjunction with the concise workbook and tutorial disk, you will find everything you need for fast-track C++ tuition.

The course is generic (i.e. compiler and hardware independent) and is available with or without the award winning Zortech C++ Compiler for MS-DOS and OS/2.

Start writing C++ code within a week.
As a C programmer, you will start producing C++ code within a week of concentrated use of this course. Alternatively, spend just an hour a day watching the video and working through the suggested exercises to learn C++ in only six weeks!

The Leader in video tutorials.
If you don't already know C, you can join tens of thousands of programmers world-wide who have learnt C with the Zortech Complete C Video Course, described as:

"An excellent bargain ... ... I heartily recommend"
Gary Ray, PC WEEK

Save your company thousands of £££'s.
One programmer can train for only £299.95, but you can train ten programmers for just:

C++ Course	*£299.95*
9 Extra Workbooks	*£179.55*
Total 10 students	*£479.50*

Yes! Only £47.95 each! (With all the FREE refresher courses you need!)

Now, Zortech with its new C++ Video Tutorial has refined the art of video tuition and presentation even further for the 1990's.

IN EUROPE CALL:

44- 1-316-7777

Only £299.95 complete
- *Six Videos with 32 lessons*
- *256 page workbook*
- *Tutorial disk*
- *Compiler & hardware independent*
- *NTSC or PAL format*
- *Tax deductible*

Don't delay, order now!
Just mail the coupon or call the order hotline for same day shipment.

USA: Zortech Inc.,
1165 Massachusetts Avenue,
ARLINGTON, MA02174
Voice: 617-646-6703
Fax: 617-643-7969

EUROPE: Zortech Ltd.,
106-108 Powis Street, LONDON
SE18 6LU
Voice: 44- 1-316-7777
Fax: 44- 1-316-4138

ORDER FORM
Please rush me these items:

Qty	*Description*	*Price*
___	*C++ Video Tutorial*	*£299.95*
___	*Extra C++ Workbooks*	*£. 19.95*
___	*Zortech C++ Compiler*	*£129.95*
___	*Developer's Edition*	*£299.95*

Please add £5 + VAT postage in UK
All UK orders please add 15% VAT
International shipping charged at cost

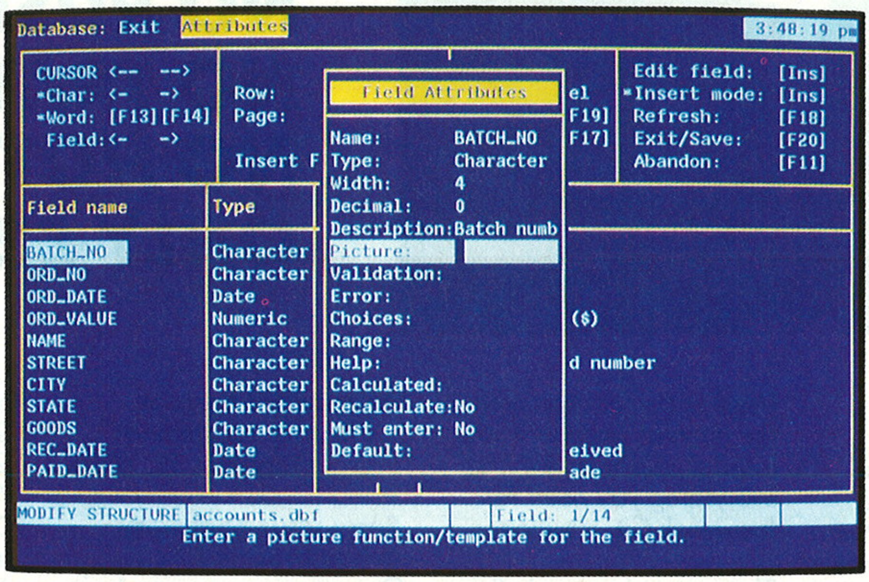
Name ______________
Company ______________
Address ______________
____
____
Phone ______________
MC,VISA or CHQ ________ Expiry ____
Card No. ______________

EXE 3/91

CIRCLE NO. 583

dBase III Plus, FoxBASE & Clipper applications running on

VAX & UNIX?



Try our 30-day software licence
In an hour or two you'll be converted

Can Recital really be that easy?

The answer is yes. And to prove it, we invite you to send for our full-featured 30-day software licence for VAX/VMS™ and UNIX systems. Recital is so compatible and easy to use, you'll have your dBase™ applications up and running in Recital in about an hour. Without any additional programming or modifications.

Then take the rest of the month to discover the richness and power of this 4GL system. Recital includes tools like a report writer, screen painter, transparent access to RMS files, an integral data dictionary, popup calendars, calculators, pick lists, note pads and much more. All on a VT-type terminal!

Your 30-day license will include the full Recital system, documentation and hotline support. How can you lose?

Order today by calling one of the numbers below. Or write to Recital Corporation.

Recital Corporation Limited
South Bank Technopark
90 London Road
London SE1 6LN, UK
Tel 071-401 2727
Fax 071-633 9617

Recital
CORPORATION

Recital Corporation Inc
85 Constitution Lane
Danvers
MA 01923, USA
Tel (508) 750 1066
Fax (508) 750 8097

All trademarks are the properties of their respective companies

CIRCLE NO. 584

ticipation of a little constructive criticism of the NDP manual. However, after I had spoken to a Microway representative, who frankly admitted that it was a dog's breakfast, and told me that it was due to be replaced in the next few weeks, I was soothed.

The main problem is that the bulk of the manual (which is an enormous, overfilled 3-ringed loose-leaf binder, equipped with rings which alternately spring open, depositing the pages in a heap on the floor, and spring shut, trapping your fingers) relates to V2.0 of the compiler, which was based on the Phar Lap tools. A few pages have been added to cover the NDP tools, but these contradict what is said in the rest of the manual and are not cross-referenced in the index. The text is also very sparse in its coverage of the C language and libraries - you certainly need K&R and other C references, plus a fair dollop of general compiler experience. There is an attitude of 'This software is for professionals, professionals should know what they are doing'.

That having been said, a lot of what is there is very good. There is a complete example of a tune playing program which hangs off the timer interrupt - a non-trivial exercise when the application runs in protected mode. Unfortunately the example requires you to have the Phar Lap assembler - which you might not have, if you are using the NDP tools - but it is certainly more complex a problem than is usually worked out in compiler manuals. There is a chapter devoted to a lucid discussion of IEEE floating point representations. And there are some wonderful nuggets, as in the coverage of the NDP linker, where the manual baldly states, '...the Microsoft variant of the [object module] scheme was simplified by IBM to make it difficult to download mainframe programs.' Wow!

Conclusion

The most impressive feature of the NDP C-386 compiler is the speed at which its object code runs. The compiler dealt with my test application like a dog snapping a wasp: one crunch, and its gone. The lack of run-time licence on the Ergo extender makes it an attractive alternative to Phar Lap based systems, although there is one important drawback. The Ergo product does not support DPMI, and there has been no announcement (as far as Microway knew) to the effect that it would. This means that the compiler and applications are (and will remain) incompatible with Windows and other utilities which use the 386's protected features. The compiler is rather old-fashioned, but it can be tied in to more modern tools.

Compilers like the NDP are the reviewer's Bain: it's difficult to get going, but the more time you spend with them, the more you come to appreciate their power. I think that this package is a good effort, and you should certainly consider it if you are looking around for something to give your code more oomph.

EXE

The NDP C-386 compiler for DOS costs £545 (this includes the NDP tools). The UNIX/XENIX version costs £750. The Microway Clearview debugger (not reviewed here) can be purchased separately for £250. The 20 MHz Cyrix 387-alike coprocessor used to perform the tests costs £250. All the above is available direct from Microway on 081 554 5466.

A good filler

No graphics library is complete without a good fill routine. Graeme Webster presents a supplement to his Super-VGA collection.

The September and October 1990 issues of .EXE carried two articles describing a Super-VGA graphics library which gave access to the 256 colour modes at 640 by 480 and higher spatial resolutions. The library's functions included switching to graphics mode, setting pixels, line drawing and even placing text on the screen - but omitted one class of routine frequently found in commercial offerings: the fill.

It is often necessary to draw closed shapes made up of straight lines - polygons - and fill them with colour. Even if the shape is bounded by curves these can be approximated by a series of straight lines. Superficially, the simplest way to fill a polygon is to examine each pixel to see if it lies inside the boundary. If it does, you set it. This is a very wasteful technique because most pixels will lie outside. The efficiency is improved by considering only pixels within the polygon's bounding box, ie the smallest rectangle containing the shape. For convex polygons, like that in Figure 1a, there can be a significant reduction in the number of pixels to be tested, but for re-entrant shapes, like that in Figure 1b, the improvement is much less.

Rather than work in pixels, a more efficient technique is to consider polygons one horizontal scan line at a time. The first step is to determine for each scan-line where, if at all, it intersects the polygon's edges, then sort the x-values of the points of intersection into ascending order. Figure 2 shows a polygon crossed by two scan-lines. The polygon will be filled correctly if a line is drawn from 1 to 2 on A-A and lines 1 to 2, 3 to 4 on B-B and so on. In other words, you just join up the sorted points of intersection in alternating pairs.

The difficulty arises from the difference between 'ideal' pixels and the blobs that appear on the screen

There are two exceptions to the rule described above. In Figure 3, scan-lines C-C and D-D illustrate one problem. C-C Intersects exactly at a local maximum. All will be in order if two intersections are counted for this point provided that the line drawing part of the algorithm can deal properly with lines of zero length (the same would apply to intersections at local minima). However, D0-D1, which intersects at a point of inflexion at D0, presents a new problem. If two intersections were to be counted there, D0-D1 would have three intersection points and the subsequent line drawing would go wrong with colour being applied to the right of D1.

The second difficulty arises from the difference between ideal mathematical pixels and the rectangular blobs that appear on the computer screen. Pixels are thought of as simple points so that, for example, a vertical line drawn from (100,100) to (100,200) would be taken to be a mathematically ideal line of zero thickness and

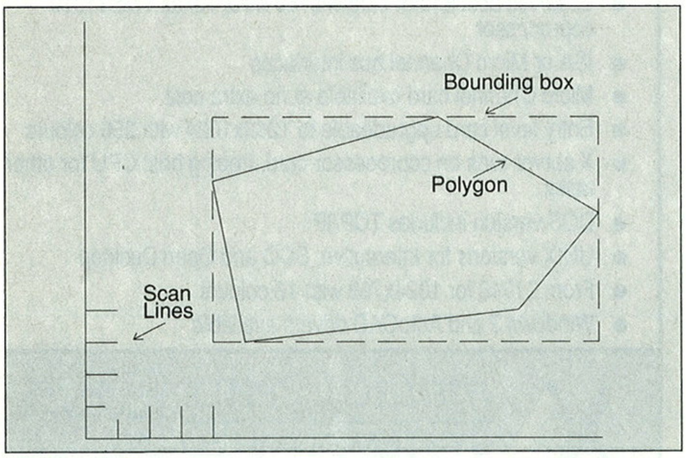


Figure 1a - Bounding boxes around polygons

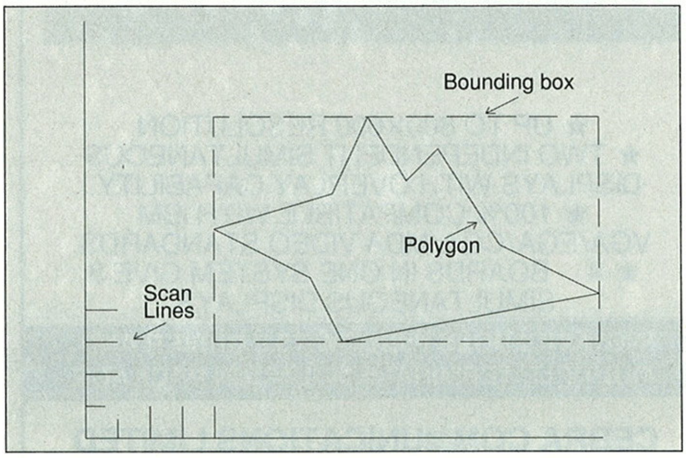


Figure 1b - Bounding boxes around polygons

"With or without a mouse, it's the cat's whiskers."

New development environments and technology transition make 80's editors inefficient and obsolete. To be competitive, programmers must have leading edge tools. The tool you'll use the most, the one that wraps everything into a single consistent environment will be the powerful, configurable SPE Professional Editor.

Right out of the box, it'll make you more productive, more adaptable, more imaginative. Its instant installation, elegant mouse support, advanced user interface, and point-and-shoot help get you running immediately.

MS-DOS, OS/2 and Dual Mode together

MS-DOS, OS/2 and Dual Mode versions are all included in the same box on both $5\frac{1}{4}''$ and $3\frac{1}{2}''$ disks, with templates for popular languages, with or without a bundled Microsoft® Mouse. Great value, and your team can move into the 90's with the same editor.

Turnkey emulations

If you prefer the commands and keystrokes of other editors, our turnkey emulations duplicate them precisely. And you still gain the SPE Professional Editor's advanced features, windowing capabilities and powerful engine.

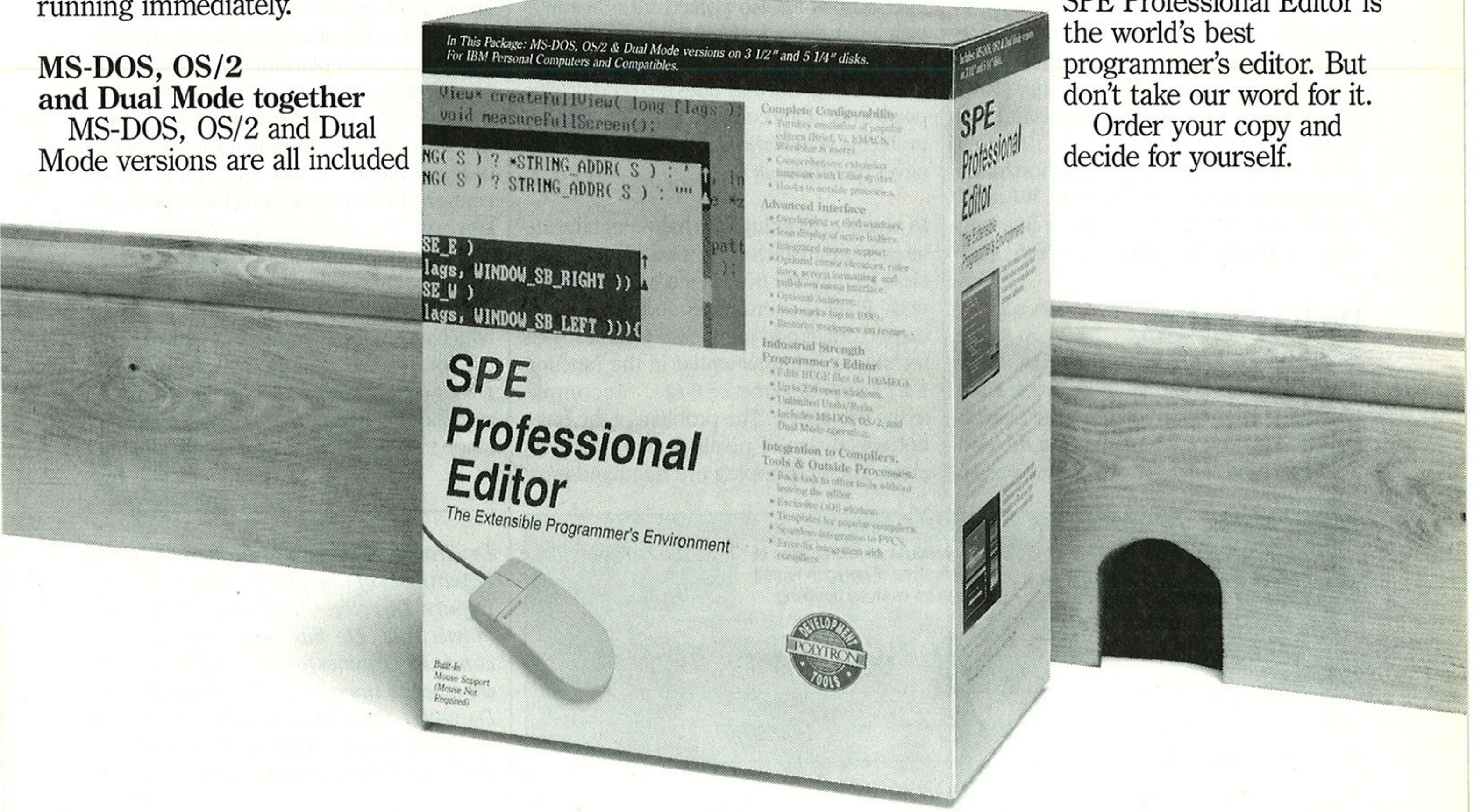
With or without a mouse

Use the editor with or without a mouse — all functions are available without lifting your fingers from the keyboard. But the click-select windows, scroll bars, zoom, shrink-to-icon, block text selects, and other speedy mouse actions make editing extremely efficient.

Don't take our word for it

We're confident that the SPE Professional Editor is the world's best programmer's editor. But don't take our word for it.

Order your copy and decide for yourself.



Works with or without a mouse.
Packaged with or without Microsoft® Mouse.

Version 1.1 now available

APPROVED DEALERS

Absolute Computing Limited, Absolute House, 124 Leavesden Road, Watford, Herts. WD2 5EG. Tel: (0923) 242570

Computer Solutions Limited, Canada Road, Byfleet, Surrey KT14 7HQ. Tel: (09323) 52744

Grey Matter Limited, Prigg Meadow, Ashburton, Devon TQ13 7DF. Tel: (0364) 53499

Roundhill Computer Systems Limited, Orchard House, Ogbourne St. George, Marlborough, Wiltshire SN8 1SU. Tel: (0672) 84535

Reflex Technology Limited, 9 Buckingham Place, Bellfield Road, High Wycombe, Bucks HP13 5HW. Tel (0494) 465907

Frontline Distribution Limited, Intec 1, Wade Road, Basingstoke, Hampshire RG24 0NE. Tel: (0256) 463344

System Science, 3-5 Cynthia Street, London N1 9JF. Tel: 071-833 1022

Software Express, Portmill House, Portmill Lane, Hitchin, Herts SG5 1DJ. Tel: (0462) 422525

Software Generation · Portmill House, Portmill Lane, Hitchin, Herts SG5 1DJ

CIRCLE NO. 587

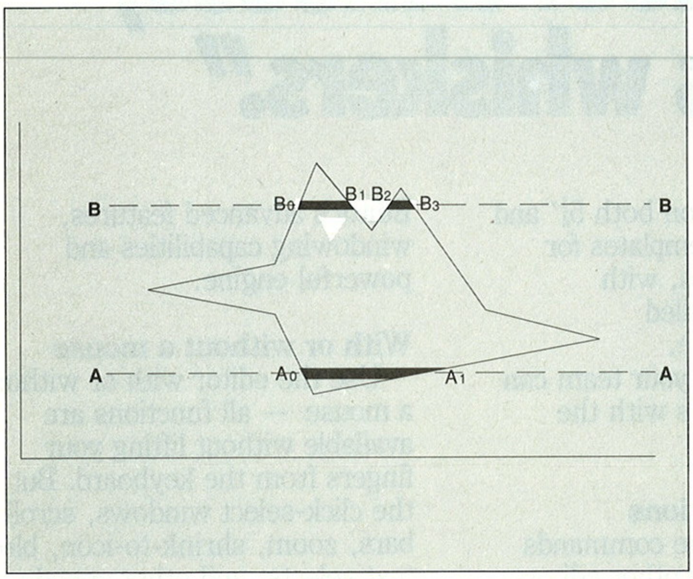


Figure 2 - Polygon crossed by scan-lines

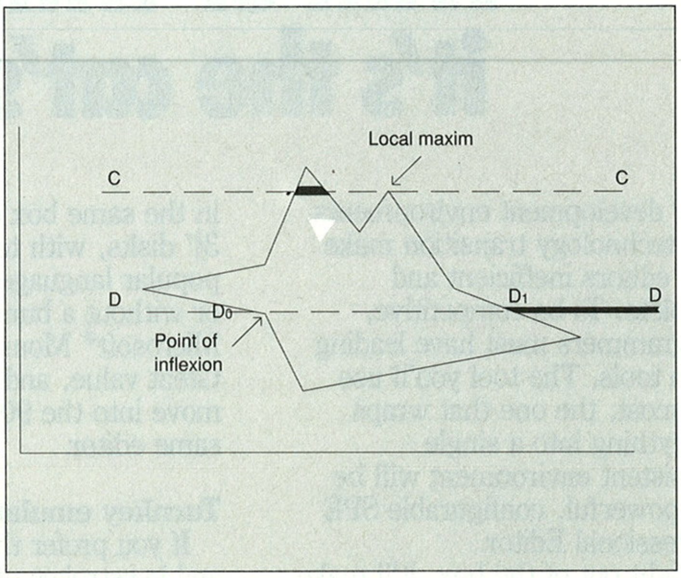


Figure 3 - Scan-lines intersecting with polygon

length 100 pixel units. In fact it is a long thin rectangle, one unit thick and 101 units long. Consider a rectangular block defined by mathematical coördinates (5,5), (15,5), (15,10), (5,10). The height of the block is five units (count it out: 5 to 6 is one unit, 6 to 7, 7 to 8, 8 to 9, 9 to 10 - a total of five units). Similarly it is 10 units wide. However, if it were to be drawn with pixels from (5,5) to (15,5), y-values from 5 to 10 would cover six scan-lines and each horizontal segment would be 11 pixels long. We need somehow to leave out one scan-line and drop one pixel horizontally, but which ones?

Do it by halves

An elegant trick which helps to solve both these problems is shown in Figure 4. First, round all vertex coördinates to the nearest integer. This is not a hardship - in the end there has to be a whole number of pixels. Then scan at half-integral y-values. If you like, you can think of it as if the scan-lines passed through the centres of the little squares containing the pixels. The x-values of the intersections are also worked out as half-integers, ie on the pixel vertical centre-lines. Finally, the pixel coördinates are rounded down before filling. Not only does this prevent extra pixels from being counted, it neatly avoids all problems with vertices, since now scan-lines never pass through them!

In practice, working with half-integral scan-lines is unsatisfactory. It is much more efficient to play the trick in a slightly different way. Pixel y-coördinates and scanning is carried out at odd numbered y-values. This technique is implemented in the function `FilledPolygon256()` accompanying this article. The problem of the finite horizontal size of pixels is dealt with less elegantly by dropping the righthand-most pixel of segments two or more pixels long.

`FilledPolygon256()` makes use of calls to the Super-VGA library, which you are strongly urged to obtain, but can easily be doctored for other graphics libraries by an appropriate replacement for the one call to each of `HorizLine256()` and `SetPixel256()`. The rest of the code provides a simple environment for testing out polygon filling - it too will require minor changes if you are not using the Super-VGA library. `FilledPolygon256()` is quite robust. It is insensitive to the ordering of vertices and will cater happily with convex and re-entrant polygons and even ones which cross over themselves. Vertical shading is easily accomplished by changing the colour parameter from line-to-line. For more complicated effects the parameter can be made to depend on the pixel coördinates.

EXE

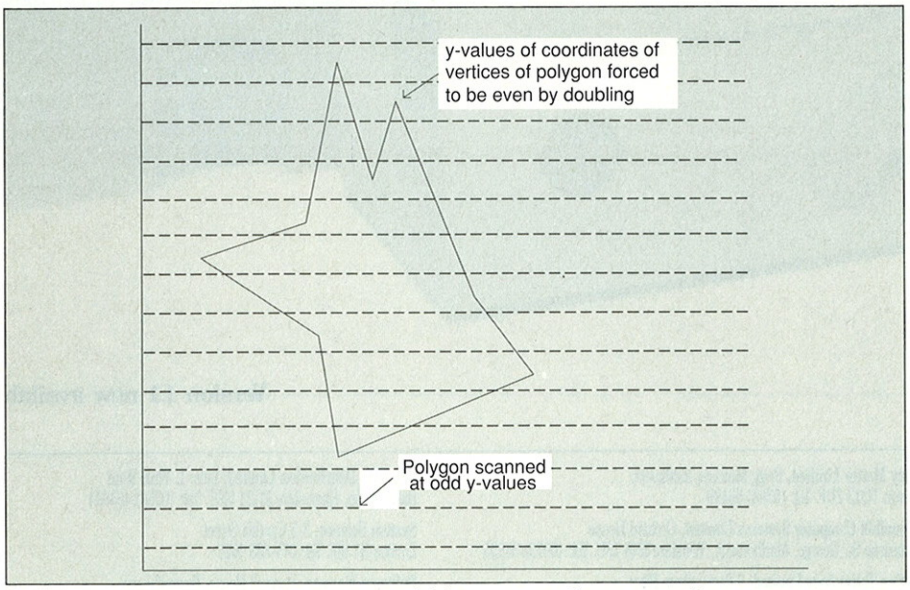


Figure 4 - Scanning an odd lines avoids even-value vertices

Dr Graeme Webster was formerly Head of Department of Computer Science and later Deputy Director, Academic, of Teesside Polytechnic. He has been involved with computer graphics for the last 20 years with an especial interest in 3D visualisation for Designers. Currently setting up a Centre for Scientific Visualisation under the aegis of the Teesside Development Corporation.

To obtain a copy of the source code for the program given in this article, together with the complete Super-VGA library from Dr Webster's earlier articles, please send a blank, formatted disk and a stamped, self-addressed envelope as described in the 'Editorial' notes on Page 1. Follow the instructions on Page 1 exactly, or we may not be able to return your disk. Mark envelopes 'SUPER VGA'.

Objects of Desire

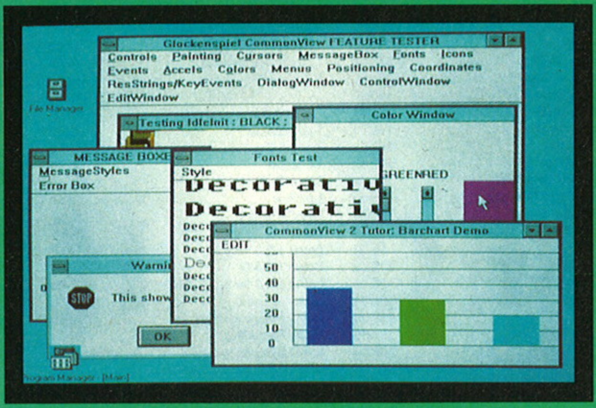
glockenspiel

CommonView 2

***Glockenspiel CommonView** has really made its mark in the world of Windows development. Thousands of developers have used it to speed up and simplify their projects.*

Now **CommonView 2** is available, inheriting the success of its predecessor and extending its capabilities even further to deliver efficient Windows 3.0 apps.

That's because **CommonView 2** works with **Glockenspiel C++ 2.0**, giving you a C++ object-based framework that reduces the complexity, cuts the code, manages memory and lets you stay in touch with what you're really doing. From compilation to execution, **CommonView 2** applications are fast and powerful.



CommonView 2 objects model the real world of Windows. And that's just the beginning - from there **CommonView 2** takes you as far as you want to go.

It doesn't anticipate everything. That's where C++ takes over: its inheritance capabilities enable you to extend the **CommonView 2** framework. Just take an existing object and customize it without affecting the original. A few additional lines of code and you could, for example, turn an edit control into a password control. Then reuse it in other applications.

You *don't* have to learn a complete new language, you *can* integrate existing developments; it *is* portable to PM and NewWave; CodeView *can* be used on your C++ source.

The world wants Windows 3.0 apps now! And with **CommonView 2** you can inherit the earth.

Specifications.

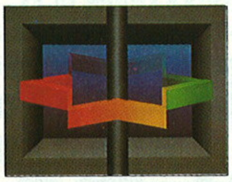
Glockenspiel CommonView 2 includes Glockenspiel C++ 2.0 and Container - the object storage framework. It requires Microsoft C 6.0, the Windows SDK and 1.5 meg of memory. You debug C++ source with Microsoft CodeView 3.0. Glockenspiel C++ supports a completely portable memory management system. Glockenspiel CommonView consists of approximately 65 classes.

Comprehensive documentation includes CommonView tutorial and reference manual, Glockenspiel C++ compiler manual and User Guide, C++ syntax and AT&T Library Guide, pullout guide to compiler switches, *plus* "Programming in C++" by Stephen C. Dewhurst and Kathy T. Stark (Prentice Hall).

On-line hypertext documentation for CommonView reference manual and AT&T guides. Tutorial source code also on disk.

Glockenspiel C++ works from within the Programmer's Workbench with the reference guides instantly available from the on-line Advisor, using Microsoft CodeView for debugging.

Glockenspiel CommonView applications are portable between Windows 2.1 and Windows 3.0, PM 1.1 and PM 1.2 with HP New Wave 3.0 version coming soon.



glockenspiel
class constructors

QA

QA Training Limited,
Cecily Hill Castle,
Cirencester, GL7 2EF.
Phone: 0285 655888. Fax: 0285 650537.

CommonView costs £495 + VAT and is available for Windows 3, OS/2 Presentation Manager and HP NewWave.

Glockenspiel Professional C++ 2.0 Compilers are available for DOS, OS/2 and Workstation platforms. Call for details.

CIRCLE NO. 588

WE GUARANTEE Y
WIDER RANGE OF SO

Over 10,000 Software Products

● COMMUNICATIONS ● DATABASES ● dBASE ADD-ON'S ● DEBUGGERS ● DEMO'S & PROTOTYPING ● DESKTOP PUBLISHING ● DIAGNOSTICS ● DOCUMENTORS ● E-MAIL● EDITORS ● EDUCA

LOTUS 1-2-3 ● 386/486 PRODUCTS ● ACCOUNTING ● C LANGUAGE ● CAD/CAM ● CASE & FLOW ● CD-ROM ● CLIP-ART ● CLIPPER

"SELLING

TO THE

@Base	145.00	Close-Up/Customer	139.00	dSalvage Professional	149.00	Greenleaf Communications	239.00	LotusWorks
386MAX Pro	79.00	Close-Up/Support	169.00	Dungeon Master	24.99	Greenleaf SuperFunctions	179.00	Mace Utilities '90
ABC Flowcharter/Windows	199.00	CO/Session	159.00	Duplicator Toolkit	69.00	Gray F/X	299.00	Manifest
Above Disk Plus	65.00	CodeBase 4	195.00	Dynacomm/Windows	279.00	HALO '88	329.00	Map Assist
Advanced Revelation	595.00	Coherent/PC	125.00	EasyCASE Plus	199.00	Harvard Graphics	279.00	MathCAD
Allways 1-2-3/Symphony	99.00	Colorix VGA Paint	125.00	Easyflow	149.00	Harvard Project Manager	495.00	MathCAD Add-On's
Alpha Four	349.00	Command Plus	79.00	Edix	129.00	hDC FistApps	59.00	Mathematica/386 w/387
Ami Pro/Windows	325.00	Control Room	79.00	EMACS/DOS	239.00	hDC Windows Express	59.00	Mavis Beacon Typing
AmiDIAG	75.00	Copy II PC	27.00	EMACS/UNIX	349.00	Headroom	75.00	Media Master Plus
Animator	269.00	Copywrite w/Zerodisk	53.00	EnerGraphic.	249.00	Hi-Screen/XL	99.00	Memory Mate
Automate/Anywhere	129.00	Corel Draw/Windows	279.00	Essential Graphics	239.00	Hijaak	89.00	Menugen
Applause II	279.00	Corporate Ladder	69.00	EXP	269.00	Hooray for Henrietta	19.95	MenuWorks
Arkanoid II	29.00	Crosstalk Mark IV	109.00	Express Publisher	129.00	Hot Wire	99.00	MetaWindow Plus
Arts & Letters/Windows	395.00	Crosstalk/Windows	97.00	Extend-a-Name Plus	73.00	Hotshot Graphics	159.00	MicroTSP
AskSam	185.00	Cruise Control	43.00	Extra-K/Symphony	69.00	HyperAccess/5	169.00	Milestones, Etc/Windows
Asyst Complete	1995.00	Dan Bricklin's Demo II	159.00	FaceLift/Windows	79.00	Hyperpad	99.00	Mirror III
ATI/Training	59.00	DASM	159.00	Family Tree Maker	49.00	IBM Current/Windows	295.00	MKS Toolkit
Auto-Might	59.00	Data Junction Advanced	229.00	Fantavision	57.00	IBM Displaywrite IV/V	299.00	Monarch
AutoImport	129.00	Dataease	429.00	Fastback Plus	99.00	IBM DOS 3.x	89.00	Move 'Em
Automenu	45.00	Dataease LAN Pack	569.00	Fastlock	47.00	IBM DOS 4.x	99.00	MS BASIC Compiler
Autoroute Plus	215.00	DataPerfect	289.00	Fastlynx	85.00	IBM Fixed Disk Organizer	47.00	MS C Compiler
AutoSketch v3	79.00	dBASE III Plus	327.00	FastText Search/Clipper	79.00	IBM OS/2 Standard Edition	239.00	MS COBOL Compiler
Avery LabelPro/Lasers	89.00	dBASE IV/Standard	369.00	FastTrax	45.00	IBM Storyboard Plus	229.00	MS Excel/Windows
B-Tree Filer	69.00	dBFast/Windows	329.00	File Rescue Plus	37.00	IBM Writing Assistant	129.00	MS Flight Simulator
Baler	310.00	dBrief	79.00	Flowcharting 3	149.00	Illustrator/Windows	289.00	MS F/S Scenery Disks
Batcom	45.00	dBXL Diamond w/R&R	149.00	Folio Views	379.00	Impress	82.00	MS FORTRAN Compiler
Battery Watch	32.00	Deluxe Paint/Enhanced	79.00	Fontasy 3 Deluxe	79.00	Individual/Training	59.00	MS Macro Assembler
BeckerTools/Windows	69.00	Derive	125.00	Fontrix	129.00	Info-XL	149.00	MS Mouse + Paint
Bitstream Fonts (each)	135.00	DesignCAD/2D	159.00	Fontspace	69.00	Info Select	85.00	MS Multiplan
Blast/PC	159.00	DesignCAD/3D	229.00	Formtool Gold	63.00	Instant Recall	95.00	MS Pascal Compiler
Blinker	199.00	Designer/Windows	399.00	Formworx/Windows	249.00	Instaplan 5000	449.00	MS Powerpoint/Windows
Blue MAX	89.00	Desk-Link	95.00	Formworx w/Fill & File	99.00	Intelligent Backup	95.00	MS Project/Windows
Brainmaker	175.00	DESKpress	229.00	FoxGraph	225.00	Kedit	109.00	MS Quick C
Brief Editor/DOS or OS/2	179.00	DESQview	79.00	FoxPro	429.00	Kermit for DOS	39.00	MS Quick Pascal
Brooklyn Bridge	79.00	DESQview/386	99.00	FoxPro/LAN	599.00	KnowledgePRO/Windows	529.00	MS QuickBASIC
Btrieve for DOS	389.00	dGE Graphics	229.00	Fractools	49.00	Labels Unlimited	42.00	MS Windows SDK
Builder	99.00	Direct Access	54.00	Framework III	339.00	Language Master	79.00	MS Windows v3
C-DOC	129.00	Direct/Net	149.00	Freedom of Press	299.00	LANSpool	249.00	MS Word
C Asynch Manager	149.00	Disk Explorer	59.00	Fresh Utilities	149.00	Lantastic Starter Kit	379.00	MS Word/Windows
C Tools Plus	109.00	Disk Manager	69.00	Fun School III	21.00	Laplink III	89.00	MS Works
Cache86	39.00	Disk Manager-N	159.00	Funcky Library	149.00	Laser-Ready	85.00	MSDOS 3.x w/GWBASIC
Calendar Creator Plus	49.00	Disk Technician Advanced	110.00	GB/Executive	169.00	Lattice C	170.00	MSDOS 4.x w/GWBASIC
Canvas/Mac	179.00	Disney Educational	21.00	GEM Artline	389.00	Le Menu	53.00	Multi-Edit Professional
Captive	30.00	Documentor	179.00	Generic 3D Drafting	329.00	LetterPerfect	139.00	Multi-Lingual Scholar
Carbon Copy+	99.00	DoDot/Windows	129.00	Generic CADD Level 3	329.00	LimSim	79.00	Multimate 4
Carbon Copy+ +Host	125.00	DOS Partner	69.00	Genifer III	199.00	LocoScript/PC	125.00	Multimate Advantage II
Carbon Copy+ AutoPilot	75.00	DosUtils	65.00	Glyphix/WordPerfect	89.00	Lotus 1-2-3 v2.2	259.00	Multiscope/Windows
Cardbox Plus	225.00	DoubleDisk	79.00	Gofer	59.00	Lotus 1-2-3 v3.1	339.00	Netlib Library
Certus/PC	179.00	DR DOS v5	110.00	GoScript Plus	235.00	Lotus 1-2-3/G	550.00	NetUtils
Charisma/Windows	299.00	Dr. Solomon's Toolkit	49.00	Grammatik IV (UK)	95.00	Lotus 1-2-3 for UNIX/386	525.00	Netware C Interfac
CheckIt	85.00	Dr. Switch	79.00	Grammatik IV/Windows	69.00	Lotus Agenda	273.00	Neuroshell
Clarion Pro Developer	549.00	Drafix CAD/Windows	495.00	Grapher	149.00	Lotus Freelance Plus	279.00	New Print Shop
Clear Plus/C or dBASE	129.00	DrawPerfect	329.00	Graphics Link Plus	139.00	Lotus Magellan	99.00	NewsMaster II
Clipper v5	325.00	DS Backup Plus	72.00	Grasp	109.00	Lotus Symphony	419.00	Norton Anti-Virus

THIS IS A SMALL SAMPLE OF OUR RANGE

WE'LL BEAT ANY NATIONAL

Software Paradise is an independent software reseller representing more than 2000 manufacturers with over 10,000 software products for IBM Personal Computers (and the Apple Macintosh) running DOS, OS/2 and UNIX. We offer the most comprehensive range of PC software in the world and have been supplying both users and programmers with quality products and the service at the lowest prices since 1986. "The Discerning Person's Guide to Software" is the most complete Buyers's Guide to PC software available and is published quarterly by Software Paradise. Call us on (+44) (0222) 887521 or complete the coupon to receive your **FREE** copy.

FREEPHONE 0800 378·873

ORDER LINE ONLY

PLACE YOUR ORDERS FREE OF CHARGE FROM ANYWHERE IN THE UK

ALL OTHER ENQUIRIES TEL: (+44) (0222) 887521 (10 LINES) FAX: (+44) (0222) 862209 (24 HRS)

OU WON'T FIND A
FTWARE ANYWHERE*

for DOS, OS/2, Mac & UNIX

AL ● EXPERT SYSTEMS & A.I ● GRAPHICS ● HARD DISK BACKUP ● INTEGRATED SOFTWARE ● LANGUAGES ● LINKERS ● MAC SOFTWARE ● MATHS & STATS ● MEMORY MANAGEMENT ● MISCELLANEOUS ● NETWORK SOFTWARE ● OCR & SCANNING ● OPERATING SYSTEMS ● PIM'S & HYPERTEXT ● PRINT SOFTWARE

129.00	Norton Backup	69.00	Pizazz Plus	79.00	RTLINK Plus	329.00	Toolbook/Windows	257.00
125.00	Norton Commander	79.00	PKZIP/PKUNZIP	54.00	Saber Menu/DOS	89.00	Top Priority	79.00
39.00	Norton Editor	49.00	PlanPerfect	235.00	Saber Menu/Windows	449.00	Touch & Go	20.00
229.00	Norton Guides C, OS/2 etc	57.00	Playroom	37.00	SCO XENIX/386 Complete	1275.00	Tree Diagrammer	65.00
345.00	Norton Utilities v5	109.00	Plink-86 Plus	349.00	Security Guardian	229.00	Turbo Asynch Plus	139.00
75.00	NoSquint	45.00	PolyBoost II	65.00	Seemore 1-2-3	65.00	Turbo C++	125.00
595.00	Nutshell Plus	189.00	PolyMake	110.00	Service Diagnostics Kit	395.00	Turbo C++ Professional	205.00
35.00	ObjectVision	299.00	PolyShell	79.00	Show Partner F/X	245.00	Turbo Pascal	79.00
55.00	Object Professional	99.00	Popdrop Plus	75.00	ShowText	149.00	Turbo Pascal Professional	149.00
45.00	OmniPage-386/Windows	529.00	Portex Professional	117.00	Sidekick Plus	97.00	Turbo Professional	79.00
45.00	Open Access III	479.00	Power BASIC	79.00	Sidetalk	79.00	TurboCAD	129.00
35.00	Opt-Tech SORT	125.00	PowerMenu	69.00	Sideways	39.00	Twist & Shout	49.00
185.00	Option Board Deluxe	105.00	Powerpak/Windows	69.00	Sigma Plot	445.00	Type Manager/Windows	79.00
499.00	Optune	59.00	PreScript Deluxe	295.00	Silvercomm Library	189.00	Typing Tutor IV Plus	39.00
129.00	ORCAD/PCB	1395.00	Presentation Task Force	249.00	Sim City	26.08	Typografica Prime	49.00
169.00	ORCAD/SDT	495.00	Primetime Personal	75.00	SitBack	62.00	UI Programmer v2	279.00
169.00	Org Plus Advanced	85.00	PrintAPlot	165.00	SiteLock w/Anti-Virus	329.00	UI Touch & Go	235.00
299.00	Overlay() for Clipper	99.00	Printcache	95.00	Smalltalk V/286	129.00	UltraScript	125.00
57.00	P.D.Queue for 1-2-3	59.00	Printer Assist	149.00	Smalltalk V/PM	295.00	UltraScript PC Plus	245.00
239.00	PageGarden	63.00	Printmaster Plus	49.00	Smarterm 240	179.00	Ultravision	79.00
229.00	Pagemaker/Mac	569.00	PrintQ	89.00	Smart System	469.00	Vcache	53.00
469.00	Pagemaker/Windows	453.00	Printrite	85.00	Smartware II	469.00	Ventura GOLD/Windows	549.00
259.00	Paradox v3.5	379.00	Procomm Plus	72.00	Soft Kicker Plus	95.00	Vermont Views	349.00
39.00	Paradox Engine	269.00	Procomm Plus w/Viewdata	107.00	Soft Type/Windows	129.00	Virex-PC or Mac	79.00
19.00	Paradox SQL Link	269.00	Professional File	169.00	Softbytes	59.00	Virus Hunter	99.00
229.00	Pathminder Plus	79.00	Professional Write	229.00	Software Bridge	85.00	ViruSafe	79.00
85.00	PC Anywhere III/IV	110.00	Professor DOS	57.00	Software Carousel	57.00	VM/386	145.00
92.00	PC Guide/Windows	429.00	Project Calc/1-2-3	139.00	Sourcer/486 w/BIOS	125.00	VolksWriter 4	125.00
109.00	PC Paintbrush IV Plus	159.00	Project Scheduler/Windows	449.00	SourcePrint	69.00	Vopt	45.00
159.00	PC Tools Deluxe	79.00	Project Workbench	795.00	SPF/PC	169.00	Watchdog	185.00
295.00	PC Write	85.00	Prokey Plus	79.00	Spinrite II	89.00	Where in Europe	39.00
349.00	PC-Browse	55.00	PVCS Professional	449.00	Sprint	135.00	Who-What-When	159.00
59.00	PC-File 5	79.00	PVCS/LAN	1595.00	SPSS/PC+	99.00	Windows Workstation	595.00
57.00	PC-Fullbak Plus	69.00	Q&A	239.00	StatGraphics	429.00	Wingz/Windows	329.00
59.00	PC-Globe	69.00	QDOS-II	49.00	Sterling Accountant Plus	329.00	Wonder Plus	59.00
275.00	PC-Guard/D Plus	137.00	QEdit TSR	79.00	Sterling Financial Controller	429.00	Word for Word Pro	99.00
75.00	PC-Kwik Power Disk	59.00	QEMM/386 w/Manifest	57.00	Stonybrook Modula-2	229.00	WordPerfect	269.00
253.00	PC-Kwik Power Pack	83.00	Qram w/Manifest	57.00	Storyboard Plus	229.00	WordPerfect Addtn'l Users	149.00
299.00	PC-Lint	89.00	Quattro Pro	269.00	Super PC-Kwik	53.00	WordPerfect Office	339.00
99.00	PDC PROLOG for DOS	199.00	Quick Schedule Plus	59.00	Superbase 4/Windows	449.00	WordPerfect/UNIX	329.00
79.00	Pegasus S/U Modules	205.00	QuickPak Professional	129.00	Supercalc 5	75.00	WordScan Plus/Windows	795.00
89.00	Pegasus Senior Modules	429.00	R&R Report Writer	135.00	SuperFonts	125.00	Wordstar 2000+	249.00
139.00	PerFORM Pro/Windows	269.00	R&R Clipper/Fox Module	49.00	Superkey	55.00	Wordstar Pro	235.00
325.00	PerFORM/GEM	159.00	R&R Code Generator	99.00	SuperPrint/Windows	149.00	XENON II MagaBlast	21.70
299.00	Personal COBOL	129.00	R:Base for DOS	379.00	SuperProject Expert	459.00	XtreeNET/ELS	179.00
289.00	Personal Measure	69.00	Rapidfile	195.00	Surfer	339.00	XtreeNET/Advanced	299.00
329.00	Personal REXX	129.00	Reader Rabbit	35.00	Switch-It	72.00	Xtree Pro	63.00
149.00	Perspective Junior	129.00	ReadRight/Windows	429.00	Systat w/Graphics	599.00	Xtree Pro Gold	83.00
79.00	Pertmaster Advance	1639.00	Reflection 2+	199.00	System Sleuth	85.00	XYwrite III Plus	269.00
185.00	PFS:First Choice	123.00	Reflection 4+	229.00	System WinSleuth/Windows	85.00	XYWrite III Plus/LAN	445.00
139.00	PFS:First Graphics	117.00	Reflex v2	179.00	Take II Manager	79.00	ZINC Library	179.00
45.00	PFS:First Publisher	125.00	Relay Gold	179.00	Talking Windows	235.00	Zortech C++ Video Course	269.00
59.00	Picture Publisher	449.00	Remote 2 Complete	129.00	Timeline v4	479.00	Zortech C++ Developer	269.00
129.00	Pixie/Windows	169.00	RM/FORTRAN	399.00	Timeworks	85.00		

CALL US FOR ANY ITEMS NOT LISTED

LY ADVERTISED PRICE+

OUR POLICY

- All prices and manufacturer's specifications are subject to change without notice.
- All prices exclude carriage and VAT.
- All orders subject to availability.
- Please specify media size when ordering.
- Goods are not offered on a trial basis.
- Goods are offered for sale subject to our standard conditions of sale which are available on request. E&OE.

* Provide written proof of any U.K. mail-order software company who supplies a wider range of IBM PC software than we offer in our catalogue within 14 days of this advertisement and get yourself a FREE copy of PC Tools Deluxe!

+ We are always delighted to be given the opportunity to beat other nationally advertised prices (at the time of placing an order) subject to the same terms and conditions of the company being beaten.

All Trademarks and Registered Trademarks are hereby acknowledged.

CIRCLE NO. 497

SOFTWARE PARADISE

The Software Experts

Software Paradise, Avenue House, King Edward Avenue, Caerphilly, Mid Glamorgan, CF8 1HE U.K.

FREEPOST — TODAY —

NO STAMP NEEDED

(U.K. ONLY)

☐ Please send me my *FREE* 32 page copy of *'The Discerning Person's Guide to Software'*

NAME

COMPANY

ADDRESS

POSTCODE TEL

Post now to: Software Paradise, FREEPOST, Avenue House King Edward Avenue, Caerphilly, CF8 1ZZ

.EXE 3/91

CIRCLE NO. 589


SUMMER
87
5·0
CLIPPER
BLINKER

```
// Polygon2.c, a program for drawing arbitrarily shaped
// filled polygons in 2D

#include <search.h>
#include <stdio.h>
#include <stdlib.h>
#include <video256.h>

#define TRUE 1
#define FALSE 0
#define MaxEdge      256L
#define MaxPoly      1024L
#define MaxVertex    4096L

struct PolyEdgeStruct
{ short StartY;
  short FinishY;
  float X;
  float DeltaX;
} PolyEdge[MaxEdge];

struct PolyStruct
{ unsigned long StartVertex;
  unsigned long FinishVertex;
  short Colour;
} Poly[MaxPoly];

struct VertexStruct
{ float X;
  float Y;
} Vertex[MaxVertex];

short HRes,VRes,CenX,CenY,MaxY,MinY,Intersection[MaxEdge];
unsigned short Type,NumEdges,NumVertex,NumXns,NumPoly;

int  CompareXns(const void *pi, const void *pj);
void FilledPolygon256(struct PolyStruct pol);
void GetData(void);
void SetUpGraphics(short *type,short *HRes,short *VRes);

main()
{ unsigned short n,pol;
  float vx,vy,vz;

  SetUpGraphics(&Type,&HRes,&VRes);
  do
  { GetData();
    if (NumPoly!=0)
    { InitGraphics256(Type, HRes);
      SetDefaultPalette256(1.6);
      for (pol=0;pol<NumPoly;pol++)
        FilledPolygon256(Poly[pol]);
      getch();
    }
    EndGraphics256();
  } while (NumPoly!=0);
  return 0;
}

int  CompareXns(const void *pi, const void *pj)
{ short i,j;
  i = *(short *)pi; j = *(short *)pj;
  if (i<j) return(-1);
  if (i>j) return(1);
  return(0);
}

void FilledPolygon256(struct PolyStruct pol)
{ unsigned short i,n,swapped,v;
  short ymin,ymax,x0,y0,x1,y1,y,dy,temp;
  float d;

// First build the edge list
// Force y-values to be even, the program will subsequently
// scan on odd lines thus avoiding the problem of scan
// lines going through vertices

  x1=Vertex[pol.FinishVertex].X;
  y1=2.0*Vertex[pol.FinishVertex].Y;
  MinY=MaxY=y1; NumEdges=0;
  for (v=pol.StartVertex;v<=pol.FinishVertex;v++)
  { x0=Vertex[v].X; y0=2.0*Vertex[v].Y;
    swapped=FALSE;
    if (y1<y0)
// Force y1>y0 for finding intersections
    { temp=x0; x0=x1; x1=temp;
      temp=y0; y0=y1; y1=temp;
      swapped=TRUE;
    }
    if (y0<MinY) MinY=y0; if (y1<MinY) MinY=y1;
    if (y0>MaxY) MaxY=y0; if (y1>MaxY) MaxY=y1;
    dy=y1-y0;
    if (dy!=0) /* skip if horiz */
    { d=((float)(x1-x0))/((float)dy);
      PolyEdge[NumEdges].StartY=y0;
      PolyEdge[NumEdges].FinishY=y1;
      PolyEdge[NumEdges].X=x0+d;
      PolyEdge[NumEdges].DeltaX=2.0*d;
// For the even/odd line trick
      if (NumEdges<MaxEdge-1)
      { NumEdges++;
      }
      else
      { DrawString256("Too many edges",0,0,15,0);
      }
    }
    if (!swapped)
    { x1=x0; y1=y0;
    }
  }

  if ((MinY<(2*VRes-1))&&(0<=MaxY))
  { MinY=max(MinY,0);
    MaxY=min(MaxY,2*VRes-2);
// Find intersections
    for (y=MinY+1;y<MaxY;y+=2)
    { NumXns=0;
      for (n=0;n<NumEdges;n++)
      { if ((PolyEdge[n].StartY<y) &&
      (y<PolyEdge[n].FinishY))
{ Intersection[NumXns]=PolyEdge[n].X;
  PolyEdge[n].X+=PolyEdge[n].DeltaX;
          NumXns++;
        }
      }
      if (NumXns>0)
// Sort and draw
      { qsort(Intersection,NumXns,2,CompareXns);
        for (i=0;i<NumXns-1;i+=2)
          if (Intersection[i]==Intersection[i+1])
            SetPixel256(Intersection[i],y>>1,pol.Colour);
          else
    HorizLine256(Intersection[i],
 Intersection[i+1]-1,
 y>>1,
 pol.Colour);
      }
    }
  }
}

void GetData(void)
{ short i,j,nv;
  unsigned char *stg;

  NumVertex=0;
  printf("Number of polygons "); scanf("%i",&NumPoly);
  if (NumPoly!=0)
  { for (i=0;i<NumPoly;i++)
    { printf("Polygon %i:\n",i);
      Poly[i].StartVertex=NumVertex;
      printf("Colour "); scanf("%i",&Poly[i].Colour);
      printf("Number of vertices "); scanf("%i",&nv);
      printf("x y of vertices:\n");
      for (j=0;j<nv;j++)
      { scanf("%f %f",
       &Vertex[NumVertex].X,
       &Vertex[NumVertex].Y);
NumVertex++;
      }
      Poly[i].FinishVertex=NumVertex-1;
    }
  }
}

void SetUpGraphics(short *type, short *HRes, short *VRes)
{ printf("Type of video adapter:\n");
  printf("0Video-7,1Paradise,2ATI Wonder,\
  3Tecmar (Tseng 3000),4SOTA,\
  5Orchid Pro designer (Tseng 4000),6EIZO ");
  scanf("%d",type);
  printf("Pixels per line\
   ");
  scanf("%d",HRes); *VRes=3*(*HRes)/4;
  if (InitGraphics256(*type, *HRes)==0)
  { EndGraphics256();
    printf("Invalid mode or resolution\n"); exit(1);
  }
  EndGraphics256();
  CenX=*HRes>>1; CenY=*VRes>>1;
}
```

Figure 5 - POLGON2.C program

Not just APL without ω⊃Δ≥!

The APL programming language attracts a small band of fiercely loyal supporters, but the uninitiated are put off by its strange character set. J, APL's new-born successor, addresses this problem and a great deal besides, as Anthony Camacho explains.

The J language was launched by Iverson Software Inc at the international conference APL90 held in Copenhagen in August 1990. J is advertised as a dialect of APL, yet it is not compatible with any version of APL. The differences between J and APL greatly simplify the problems of implementation, and increase the expressive power of the notation. This article is not a tutorial in J (there isn't space) but a survey of some of its more interesting features.

APL is notorious for using a special character set. This is difficult to display and print, and also causes keyboard problems. J, on the other hand, is pure ASCII. It can be used with practically any display and printer, and you can even write J on your MicroWriter or AgendA and download it to your target machine.

Background

To understand J, we must first examine its APL background. APL was invented by Dr KE Iverson as a system of mathematical notation in the late 1950s, well before it became practical to implement it on a computer. During this 'handwritten' period, many improvements and simplifications were made. It is the only computer language which has benefited from years of manual use before implementation on a machine set its features in concrete and inhibited further improvement. For example, APL was used to model the IBM 360 computer design.

One of the motives in devising APL was a dislike of the anomalies and inconsistencies of conventional mathematical notation. Functions with a single argument are sometimes written on the left of the argument (*log x*), sometimes on the right (*x!*) and sometimes on both sides (|*x*|). APL is remarkably consistent, but a few anomalies did slip through.

APL and J are both designed to be read like English, with the main verb early in the sentence

J dispenses with all known APL inconsistencies. Dr Iverson (still working on the case!) chose not to make J compatible with APL because that would mean retaining APL's oddities. For example, to obtain the item from the 2nd plane, 3rd row and 9th column of a three dimensional array `REPT`, in APL, you must write `REPT[2;3;9]`. What are these brackets and semicolons? They clearly represent a kind of a function, but why are there *four* symbols, and why spread them through the argument like that? There is no need to preserve such an anomaly: in J there is an indexing function (`{`) with a left argument to specify the index and a right argument to specify the array. In APL, the index origin can be set at 0 or 1, so the top left corner of a matrix could be `MAT[0;0]` or `MAT[1;1]`. In index origin `0 REPT[2;3;9]` would address the 3rd plane 4th row and 10th column. To discover which alternative is meant the APL reader has to find the current value of the index origin. The J reader *knows* what `(<1 2 8){REPT` means because origin zero is used throughout.

It's English

J, like APL, is interpreted and functional. Both are designed to be read like English, with the main verb early in the sentence. (Of course, J is like all programming languages in that all its sentences are commands.) So `+/*/table` is read as *(take) the sum of the products of the elements in each item (row) of the variable* `table`. 'Sum' is the main verb. The phrase +/ (sum

```
#define RHS    (NOUN+VERB+ADV+CONJ)
#define EDGE   (MARK+ASGN+LPAR)

static struct {I c[4] ;AF f;I b.e; }cases[] ={

EDGE+ADV+VERB,          VERB,        NOUN,    ANY,          verb,1,    2,
CONJ,                   NOUN,        VERB,    NOUN,         verb,2,    3,
EDGE+ADV+VERB+NOUN,     NOUN,        VERB,    NOUN,         verb,1,    3,
EDGE+ADV+VERB+NOUN,     NOUN+VERB,   ADV,     ANY,          adv,1,     2,
EDGE+ADV+VERB+NOUN,     NOUN+VERB,   CONJ,    NOUN+VERB,    conj,1,    3,
EDGE+ADV+VERB+NOUN,     VERB,        VERB,    VERB,         form,1,    3,
EDGE,                   VERB,        VERB,    ANY,          form,1,    2,
NAME                    ASGN,        RHS,     ANY,          is,  0,    2,
LPAR,                   RHS,         RPAR,    ANY,          punc,0,    2,
ANY,                    ANY,         ANY,     ANY,          move,0,   -1,
};'
```

Figure 1 - The parsing rules in C

of) applies to its right argument, whether the right argument is a simple row of numbers or an expression giving a row or rows as its result. `*/+/table` *(take the product of the sums of the items in* `table`*)* makes 'product' the main verb. The order of execution in a J sentence can be controlled with parentheses just as in mathematics.

We have shown that J's primitive functions can be thought of as verbs. The analogy is extended further. Its constants behave like nouns, defined functions are pro-verbs, and variable names are pro-nouns. In standard APL, there are operators which take verbs as arguments and give a modified verb as a result. For example, in the phrase +/ (valid in both J and APL), the / is an operator which converts *plus* to *sum of.* What APL calls 'operators', J calls 'adverbs'. J also has conjunctions such as *with* (&) and *after* or *atop* (@). The phrase `x^2` raises `x` to the power of 2. One can define a `cube` function as *power with 3* (`cube=.^&3`). J allows pro-adverbs and pro-conjunctions to be defined by the user.

As in English, these pro-nouns, pro-verbs, pro-adverbs and pro-conjunctions rank in every way equal to nouns, verbs, adverbs and conjunctions respectively. The parser does not have to distinguish between the defined and primitive parts of speech. It is easy to say this, less easy to do it. In standard APL, for example, you cannot modify a user-defined function with an operator (adverb), there are no user-definable adverbs and no conjunctions at all.

Tokens and punctuation

The primitive elements (or tokens) of J are all composed of one or two ASCII characters, of which the second, if there are two, must be a full stop or a colon. The process of parsing involves recognising the following:

- Constants (nouns) as a sequence of characters in quotation marks, or as a sequence of digits plus decimal point, E or e and j (J includes complex numbers),
- Tokens - names of verbs, adverbs or conjunctions,
- Names (pro-nouns, verbs, adverbs or conjunctions), and
- Punctuation - matching pairs of parentheses are the only form of punctuation.

Spaces are only needed in two circumstances: first to separate names from each other and from numeric constants, and second to separate the atoms of numeric constants (so `2 3 4 5` is a one dimensional array of four numbers).

One consequence of this improved consistency is that the parser for J (written in C) is a single page of code (see Figure 1). A sentence to be parsed is placed on a left stack and, as execution proceeds, words are moved from the tail of the left stack to the front of a right stack. In general, when the first four words of the right stack match a pattern (columns 0 to 3 of the table), the corresponding action (4) is triggered and applied to the indicated words (5, 6), with the result replacing these words. The modifications applied to APL to make J have simplified the interpreter, not complicated it.

Functions

APL had more primitive functions than any previous computer language. It extends simple arithmetic with maximum, minimum, ceiling (next integer above) and floor (next integer below). It includes the beta and gamma functions in a complete set of transcendentals. It provides many matrix operations. Its encode and decode functions convert from decimal to any multi-radix representation and back. It contains the complete bit-wise propositional logic functions. It can rotate, reverse, transpose and reshape arrays.

J has a superset of APL primitives. Many functions are not exact matches to anything in APL, some are entirely new (eg functions for the determinant and permanent of square matrices, and for the product and division of polynomials). Where there is a difference of implementation, the J expression will usually be recognised as more elegant than the APL expression.

The J variable is fundamentally an array. Simple arrays may have as many dimensions as you wish, and at different times may hold characters, booleans, integers, real numbers or complex numbers. Simple arrays are rectangular and a special function, the shape function, returns the number of planes, rows and columns of an array right argument. A set of tables will have three integers in its shape. The number of integers in the shape is the rank of the array. There is no explicit limit to the number of dimensions an array may have.

To be able to apply a function to an array (or between two arrays), you must be able to determine how to select the rank of each argument the function is to work on. In J, every verb has a specification to show which ranks of its argument(s) it will work on by default. Since all pro-verbs are made of verbs, this ensures that a default rank is defined for every function. These defaults can be changed by the use of the rank adverbs.

All arrays are rectangular - ie you cannot have one row of a table longer or shorter than another. Rectangularity, imposed on every data type, wastes space or causes inconvenience. J and some advanced implementations of APL overcome the problem by providing box (<) and open (>) verbs. Any array can be boxed, and the result is an atom which can be put in any position in any other array. In standard APL, a book could be held as a rank three array of pages, lines and columns (probably with thousands of spaces filling every page to the same size as the largest). This can also be done in J - but boxing offers an alternative. The letters of each word can be boxed; then the words in each sentence can be boxed; then the sentences of each paragraph can be boxed; then the paragraphs of each chapter can be boxed; and finally the whole book could be represented as a row of chapters with the contents and appendix as special cases of a chapter. Such boxed arrays can be processed almost as easily as the simple rectangular arrays and much more efficiently. J holds sparse matrices much more economically than standard APL could.

Where the atoms in standard programming language arrays are restricted to a single type of variable (all atoms of an array must be either character, or boolean, or integer etc), J's box function allows mixed types. The headings of a report can be held in the same array as the numeric values that make up its body.

That triangle

My J example is, of course, the *Third Side* problem. I have found it hard to match the other solutions exactly. Although J can read from a file, there is nothing corresponding to `INPUT` in BASIC or to *quad* and *quote-quad*, the input functions in APL. Furthermore, the problem spec asks the author to show how looping is managed in the language being demonstrated. There are usually no loops in well-written J. The whole *Third Side* program is just one line of code plus a variable with the descriptions in it.

The expression determines the type of triangle should give a result that selects the wording to be shown. If we produce a phrase which returns a 1 for a possible triangle and a 0 otherwise, then we can multiply the result of that by the number of sides that are different. The expression will yield 0 for impossible triangles, 1 for equi-

laterals, 2 for isosceles and 3 for scalene. This can be used to index the variable `types` holding the four descriptions:

```
types=.' is not a triangle';
       ' is equilateral';
       ' is isosceles';
       ' is scalene'
```

The `=.` assigns the value. The quotes punctuate strings. The semicolons link the strings, boxing any string that isn't boxed. The result is a row of four atoms, each a box containing a description. `0{types` would return `'is not a triangle'` in a box. The open verb (`>`) does the unboxing. The expression `>1{types` would return `'is equilateral.'`

To construct a phrase that returns a 1 if the three sides could make a triangle, we test whether each side is less than half the perimeter. If all do, then we return a 1. `(%&2)@(+/)` is 'divide (`%`) with (`&`) two (`2`) after (or atop) (`@`) sum of (`+/`)' and returns the semi-perimeter (assuming its argument is a row of three side lengths). The conjunctions (`&`) and (`@`) turn the expression into a single pro-verb. Preceding this with less-than (`<`) makes a *hook*. A hook is a pair of verbs which, written 'verb verb argument' are to be interpreted as argument verb verb argument. This particular hook tests whether the three sides are less than the semi-perimeter returning three boolean values (1 when the side was less than the semi-perimeter and 0 when it was not). Then we must 'and' these three booleans together because only if they are all one is there a possible triangle. 'And' is `*.` and `/` applies it between the booleans. The complete test for a possible triangle is `test=.*./&(<%&2@(+/))`.

The type of triangle is given by the count of the number of different sides. J gives us the verb *nub* (`~.`) which returns all the different values in the right argument, and *tally* (`#`) which counts the values in its argument. 'Tally after nub', given an argument of a number of side lengths, returns the number of sides that are different. Tally after nub (`#@~.`) is one verb - call it `diffs=.#@~.` - 'times' (`*`) is a second and `test` is a third. The three form a fork which is (in our case) 'verb verb verb argument'. This idiom is interpreted as '(verb argument) verb (verb argument)'. The outer expressions in brackets give results which become the arguments of the central verb. So we can write `(diffs * test)` or `(#@~.*(*./&(<%&2@(+/))))` and give it a right argument of three side lengths to obtain a single digit result of 0, 1, 2 or 3. If we precede this expression with `({&types)@`, the result will be (for three sides) one of the three descriptions of triangles, but it will still be boxed.

Making it pretty

To open the box we need to precede the whole expression with 'open with' (`>&`). This almost completes the task, but it would be prettier if the expression would work on larger arrays such as a hundred rows of three sides. J is, after all, supposed to be good at array processing and I did say that explicit loops were to be avoided. Second, if we are to process lots of triangles at a time, it would be easier to read the answer if each triangle were printed out before its description. So you see why the descriptions all begin with 'is'.

The descriptions are strings of characters. To precede them with numbers (in an array without boxes) we have to *format* (`":`) the numbers into characters. Then we must append the descriptions onto the character-form numbers. 'Format', 'append' and the

long expression above conveniently make another fork. When followed by the three side lengths, the lengths will be formatted to characters and then appended with the description appropriate to those lengths. Finally, we want the whole expression to be applied to rank one (`"1`) of the argument, so that if the argument is a table, each row will be processed as one argument. The whole expression can be assigned to a pro-verb name such as `ts` (for 'Third Side').

The data for `ts` to work on can be expressed as `9 9 9, 9 8 8, 9 8 7, 9 4 4, 4 5 6, 4 5 5, 4 4 4,: 4 2 2` because the effect of the comma colon is to cause the last two groups of three side lengths to be appended (`,` is append) above each other and all the previous groups are then appended above them, so forming a table of eight rows of three. Isn't it convenient that the commas split the numbers into rows and make the expression easier to read at the same time? The complete program, with a sample run, is presented in Figure 2.

```
types=.' is not a triangle';' is equilateral';' is isosceles';
                                                 ' is scalene'

test=.*./&(<%&2 @ (+/))
diffs=.#@~.
ts=.(":,>&({&types) @ (diffs * test))"1

or, written out in full,

ts=.(":,>&({&types) @ (#@~.*(*./&(<%&2 @ (+/)))))"1

Sample Run
   ts 9 9 9, 9 8 8, 9 8 7, 9 4 4, 4 5 6, 4 5 5, 4 4 4,: 4 2 2
9 9 9 is equilateral
9 8 8 is isosceles
9 8 7 is scalene
9 4 4 is not a triangle
4 5 6 is scalene
4 5 5 is isosceles
4 4 4 is equilateral
4 2 2 is not a triangle
```

Figure 2 - Triangle solution and sample run

Final notes

J has been ported to PC clones (8088, 8086, 80286, 80386 and 80486), Macintosh, AT&T 3B1, IBM R6000, MIPS R3000, NeXT, Philips P9070, Sun 3, Sun 386i and Sun 4 (SPARC). An Archimedes port is in the pipeline.

J is available in the UK from I-APL Ltd. We have PC and Macintosh versions in stock. J is shareware, and you may make copies of the disk and give it away. It is still not quite complete, and the disk contains a STATUS.DOC file listing what is and isn't implemented. Order forms and leaflets can be had from: I-APL Ltd, 2 Blenheim Road, St Albans, Herts AL1 4NR.

EXE

Anthony Camacho is a freelance APL programmer specialising in fixed-price PC work. In his spare time he is Hon Sec of the British APL Association and Chairman of I-APL Ltd, the European end of a project which aims to increase the use of APL in education by issuing a free APL interpreter.

The Germans do it differently

Far from being chaotic, computer hacking in Germany is a well-organised activity. WW Osterhage has been looking into the activities of CCC - the Chaos Computer Club.

There is nothing technically special about hacking in Germany. As a pastime, it is indistinguishable from similar clandestine activities carried out in other European countries and the US. People enter databases in banks, public services, industrial companies (military hardware suppliers are especially popular) and public or private nets. Occasionally, a case makes it into the headlines.

German hacking, however, has one attribute which it shares with many other activities in this country: it is well-organised. As long ago as 1981, there was a group of lone hackers in Berlin. They used to hack away hour after hour and night after night - sometimes spending months trying to crack a single target - travelling down the forbidden paths in off-limits domains. They got together and decided that their hacking would become more efficient if their skills were combined; so they founded a registered association: the Chaos Computer Club eV, or 'CCC'.

The CCC has been on front pages more than once. Famous cases include break-ins into NASA, CERN, Philips and Btx (a German Telecom service). In some of the cases the CCC admitted semi-official involvement, sometimes the guilty individuals turned out to be CCC members, occasionally there appeared to be no connection.

CCC claims to be more than just a sporting team trying to outwit the smartest security barriers in computing systems, or discovering, say, loopholes in the VMS operating system. The Club has a charter, and stands for a certain philosophy (sometimes bordering on ideology - another German pastime). Its 200 odd members (including about 30 activists) subscribe to the following code of conduct:

- No hacking for commercial gains.
- Condemn the crasher.
- Publicise loopholes and security gaps that have been found in sensitive systems and nets.
- No alteration of data.
- Help to achieve a high level of security only for the most sensitive information (that relating to private individuals and research results).
- Help to avoid hacks by persuading the public and the influential to make as much information as possible generally available.
- Promote the free exchange of information.

But the myth of the lone hacker lingers on. In fact, when sifting through the main publications of CCC, it becomes apparent that it is nurtured. There is a semi-regular 4-8 page bulletin *Datenschleuder* ('Data catapult'), and the club's famous two-volume *Hacker's Bible.*

Hacker's Bible

These publications leave you in no doubt as to CCC's political sympathies. The Hacker's Bible (which includes many extracts from past issues of Datenschleuder) contains a remarkable synthesis of political ideas; a combination of alternative life-styles, leftist ideology, pop culture and serious high tech. It is only loosely structured and covers a very diverse range of subjects.

Features include contributions about the current hacking scene, reports describing successful hacks and other pieces of self-promotion. There are interviews with various personalities who are considered able to make a contribution to data security, information strategies or social perceptions of IT; and with great opinion shapers (eg Prof Weizenbaum, Timothy Leary).

The Bible also contains the proceedings of hackers' conferences organised by CCC. But the main (and most interesting) part of the journal represents a collection of procedures, hints and tips helpful to hacking. Sometimes the messages are hidden, cloaked in cartoons or in a funny story, sometimes they are presented as someone else's information, only quoted here. Occasionally they are preceded by pseudo-warnings as a protection against possible prosecution. Legally obtainable information is described in detail openly. Subjects covered include wiring diagrams for communication equipment, internal documents from PTTs, AT&T etc, interesting communication frequencies in use (satellites, radio etc), code proposals (including viral), protocol descriptions, extracts from data security protection documents and so on.

Still, true to its philosophy, CCC also enjoys a poacher-turned-gamekeeper role by marketing a software security product called PC-DES!

The CCC is taken seriously these days - and not only because it is seen as a threat in some quarters. Because of its accumulated store of technical expertise, and also because of some of the positive aspects of its ethics and its influence on society, its consultancy has become sought-after by serious people. Both NASA and DEC requested its advice when soft-spots in their systems had been discovered. There is exchange of opinion and information between the CCC and the public data security official of the city of Hamburg (where CCC's headquarters are). Now industrial companies are deliberately letting the hackers in - to lecture and consult on data security.

EXE

Wolfgang W Osterhage holds PhDs in Physics and Information Science. He works as an independent consultant for industry, specialising in IT and logistics. He can be contacted at Birkenweg 7, D-5307 Wachtberg - Niederbachem, Germany.

Getting in a state

State Transition Diagrams are more than just pretty pictures on a CASE tool screen. Ray Jones shows how they can be useful in the implementation of real software.

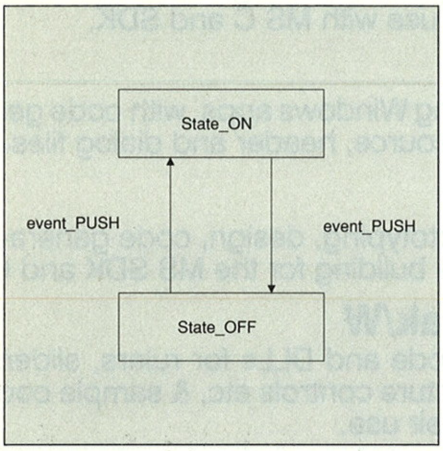


Figure 1 - STD for a mains

Those familiar with the ISO communications protocols or who have studied structured design for real time systems may also be familiar with *state machines* - or more formally, Finite State Machines - and their representation as State Transition Diagrams (STDs) or state tables. This article presents a neat way of coding state machines using a general purpose (and thus reusable) state table interpreter.

For those not acquainted with state machines a brief introduction follows The rest of you can skip to the nitty gritty C code.

What is a State Machine?

State machines are a very useful tool for the implementation of a variety of control-type problems. As the name Finite State Machine suggests, they are mechanisms that, at any one time, can be in one of a limited number of states; the state may only change in response to specific external events. The easiest way to describe state machines is with an example. Consider a push-button mains switch: the switch can be in one of two states, on or off, and it changes from one state to the other in response to a particular external event - someone pushing the button. State machines don't come much simpler than that, two states and a single event. Notice that the effect of the event depends on the state - if the state is 'off' then the event causes a change to 'on', if the state had been 'on' then the same event would have caused a change to 'off'.

As promised, the code and the technique are reusable

State machines may be represented graphically in the form of a State Transition Diagram (STD); these consist circles or rectangles (the states) and arcs joining them (the transitions caused by events), the events are shown as labels on the arcs. Figure 1 is the STD for the push-button switch.

The STD is a graphical representation of the state machine; to make it useful in software terms, it must also describe a software implementation of the machine. This is easily achieved by the addition of one extra element, each transition is labelled with an action. To implement the push-button two actions are needed, 'turn current on' and 'turn current off' - I won't insult you by specifying which transitions they belong to. I hope that you can see from this that in order to implement a state machine in software all that is needed is to code the actions and the state/transition mechanism itself.

The Nitty Gritty

OK, that's enough waffle, let's get down to the real thing - here comes the code.

An STD is not much use to a C program, so the first thing that must be done is to translate it into something more program-friendly. That something is a state table.

A state table is an array of structures, at least one for each state. Each structure contains four pieces of information, the current state, an event that is valid in the state, the action to be taken on the occurrence of that event and, finally, the state to which a transition is made following the event. (To give you the idea, Figure 2 is a state table equivalent of the STD in Figure 1.)

Apart from the state table itself, we need some actions, a mechanism for grabbing events and the state table interpreter. All this is readily coded in C (and most other languages, too). The example that I've written is in C, is entitled 'States of Mind', and purports to analyse your current mental condition (it's probably best not to try it out right now). The STD for 'States of Mind' is given in Figure 3.

The code is shown in Figure 4; a little explanation is, perhaps, needed. The state table is the defined type `STATE_TABLE`. The state, event and next state parts are enumerated types, the action part is a reference to the function that implements the required action. C can use this reference to

Current State	Event	Action	Next State
State 'on'	Push	Switch current off	State 'off'
State 'off'	Push	Switch current on	State 'on'

Figure 2 - State Table equivalent of Figure 1

call the action routine directly; in Pascal this, too. would have to be an enumerated type and the correct function would be called via a `case` statement. Note that the action routines must all be of the same type (ie must return the same type of value) and can have no parameters passed to them.

The state table interpreter is fairly straightforward. It waits for an event, and having received one, searches through the state table entries that match the current state looking for the event that has just been received. If it is found, the appropriate action is called, and the current state is updated with the next state value for this state/event. The interpreter then loops around with the new state value to wait for the next event.

Making it Better?

Other implementations of both the state table and state table interpreter are possible. For example, the table could be compressed by making its structure a little more complex. As implemented here the state table is an array of simple structures, a more efficient representation might have only one structure per state with the multiple event and next state parts being in an embedded array of structures - this may be useful if there are many events that are relevant to each state - the interpreter would also have to be modified slightly to cater for this.

As promised, the code and (perhaps more important) the technique are reusable. All you have to do is code up the actions and the event gathering mechanism and you're away. If you ever have to put together an ANSI terminal emulator (may God preserve you), you'll discover that, when it comes to handling escape codes, this technique is the biz.

EXE

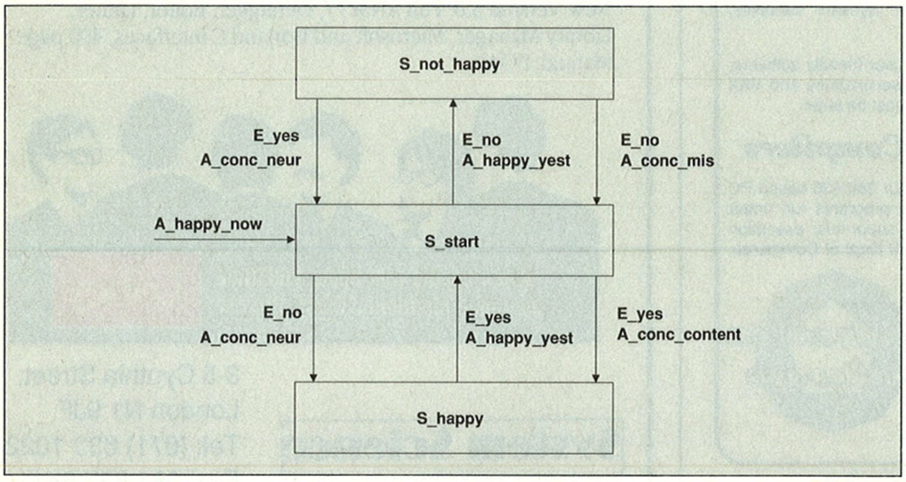


Figure 3 - States of Mind STD

Ray Jones is a software engineer who has been designing and implementing technical and real time software for so long that he remembers when all microprocessors were RISC machines.

WITH HIGHLAND ADD THE POWER OF GRAPHICS!

DYNAMITE DEMOS with GREAT GRAPHICS

Prototype & Demo System

Now you can demo your applications with Highland's PictureBox in almost any graphics mode including SuperVGA 800x600x256 or any of the standard VGA modes.

Use the integrated development system and the powerful Zsoft PC PaintBrush IV Plus included to create, organize, edit, and debug your demo.

Use special effects like Wipes, Explodes, Fades, Splits, Diagonals, etc. on the PCX bitmaps or on the text screens.

All commands can be controlled through mouse driven 3D button menus for intuitive and intelligent program design.

No Royalties Provided is a free runtime version you can distribute with your professional demo.

+More

Supports 30+ graphic modes plus text, with special effects; cut, move, size & scale images; text on graphics; simple to use - no programming; 250+ page manual with tutorial; build fully interactive demos that respond to keystrokes, data entry, timeouts and mouse events, and much more.

PictureBox Only!

£149.95

Order Today!

PCX *Effects & Text* IMAGE Solutions!

VBase Colour Database £249.95

Turn your dBase, Clipper, C, Basic, Pascal, application into a powerful 256 - COLOUR DATABASE with little programming skills. Includes full printer support ! Most any application can be converted in just a few hours.

PCX Programmers Toolkit £165.00

Includes over 75 routines to display, save, scale, and print PCX bitmapped graphics from almost any programme. Full use of virtual buffers and EMS 4.0.

PCX Effects £85.00

Add spectacular special effects (F/X) to your program. Wipe, split, crush, slide, sand, drip, diagonal, spiral, random, or explode your graphics. Requires PCX Toolkit.

PCX Text £149.00

Display blazing bitmapped text in any graphics mode as simple as text mode. You can display your text information on top of graphics.

GX Graphics £179.95

A complete graphics library supporting all graphics primitives. Use the GX Graphics Toolkit instead of the BGI or MS libraries and make your program faster, smaller and more portable across compilers - while supporting more video modes.

Pinna Electronics Ltd., APL Centre, Stevenston, Ayrshire KA20 3LR

Tel: (0294) 605296 or Fax: (0294) 68286

CIRCLE NO. 607

System Science

C COMPILERS	
Microsoft C 6.0 with PWB	£245.00
Quick C & Quick Asm 2.51	£99.00
Quick C ver 2.5	£65.00
Turbo C++ Professional	£199.00
Zortech C++ Dev Ed	£275.00
Watcom C 8.0	£255.00
Aztec C86 Comm	£265.00
C compilers for 386	£call

C GRAPHICS & DATAFILE	
Halo	£195.00
GSS XVT for Windows	£345.00
Metawindows	£175.00
Essential Graphics (no roy)	£275.00
PCX Prog Toolkit (new 4.0)	£165.00

C DATA & MISC	
Btrieve for DOS	£345.00
XQL for Btrieve	£475.00
Ctree (source)	£195.00
CodeBase 4.2 (source)	£175.00
db_VISTA III	£545.00
GL Financial Mathlib	£245.00
Blaise C Tools Plus	£89.00

C SCREENS & COMMS	
Cscape, Look & Feel	£370.00
Vermont Views & Designer	£295.00
Panel Plus II	£275.00
Aspen Formation (new ver)	£195.00
C Worthy & cwARCHITECT	£call
Blaise C Aynch Manager	£115.00
Greenleaf CommLib	£185.00

PROGRAMMERS TOOLS	
MKS Toolkit (Unix toolset)	£155.00
MKS RCS (Rev Cont Sys)	£125.00
PVCS Prof (lower price)	£375.00
PolyMake	£135.00
vsDesigner (Case Tool)	£475.00
Brief Editor 3.0	£195.00
Kedit Editor	£110.00
Flow Charting 3	£175.00
Dan Bricklin DEMO II	£195.00
MS Progs Lib 1.2 CD	£225.00
Personal Rexx	£110.00

ASSEMBLERS & LINKERS	
MS Macro ASM 5.1	£95.00
Turbo Debugger & ASM 2.0	£99.00
RTLink Plus	£345.00
Plink 86 Plus	£295.00
Sourcer & Bios (DisAsm)	£125.00

CROSS DEVELOPMENT	
Introl C or Modula-2	£1445.00
Aztec C Cross	£275.00
IAR C Cross with CSpy	£call
2500AD Cross AsM	£135.00
Simulators & others	£call

FORTRAN	
MS Fortran-77 5.0	£235.00
Lahey F77L-EM/32 & 386 OS	£795.00
Salford FTN77/386	£765.00
Lahey Personal Fortran	£79.00
InGraf for Fortran (source)	£175.00

PASCAL, BASIC	
Turbo Pascal (6.0)	£75.00
Object Professional (for TP)	£110.00
MS Quick Pascal	£60.00
Blaise Power Tools Plus	£89.00
Blaise Asynch Plus	£89.00
MS Basic 7.1 with PWB	£255.00
MS Quick Basic 4.5	£65.00
PowerBasic (ex Turbo Basic)	£85.00
many Pascal, Basic libraries & tools	£call

OTHER LANGUAGES	
Smalltalk V	£69.00
TopSpeed Modula-2	£145.00
MS Cobol 4.0 (new ver)	£445.00
RM Cobol-85 ver 5.0	£995.00
LM1 or Forth	£225.00
PC Logo	£50.00
Clipper 5.0	£395.00

MEMORY, MATHS etc	
386 Max 5.0	£75.00
QEMM/386 5.1	£60.00
QRAM for 286	£55.00
Above Disc (LIM Emul)	£69.00
Norton Utilities (new ver 5)	£110.00
Spin Rite II	£69.00
Derive (Symbol Alg) 2.0	£135.00
MicroStat II	£295.00
Mathematica by Wolfram	£call

WINDOWS 3.0	
MS Windows 3.0	£85.00
MS Windows 3.0 & Mouse	£125.00
MS Windows SDK 3.0	£325.00
Actor 3.0 with Lang Ext	£625.00
WinPro/3 from Xian	£525.00
WindowsMaker	£595.00
Toolbook by Asymetrix	£295.00

"OUR RANGE INCLUDES MANY MORE ITEMS FOR DOS, OS/2, UNIX ETC. PLEASE ENQUIRE"

3-5 Cynthia Street, London N1 9LF
Fax: 071-837 6411

BARCLAYCARD VISA · Access

Prices are subject to change and alteration Prices include UK Delivery but are exclusive of VAT

071-833 1022

CIRCLE NO. 608

```
/* State table interpreter  */
/* Ray Jones for .EXE */

#include <stdio.h>

/*  States and Events */

enum T_state { S_start, S_happy, S_not_happy,
               S_last };
enum T_event { E_yes, E_no };

/* Actions - they must all be functions of the
   same type and take no parameters */

void A_happy_today()
{
   printf("Are you happy today?\n");
}
void A_happy_yest()
{
   printf("Were you happy yesterday?\n");
}
void A_conc_content()
{
   printf("\nYou are a content person.\n\n");
   A_happy_today();
}
void A_conc_mis()
{
   printf("\nYou are a miserable b*gger.\n\n");
   A_happy_today();
}

void A_conc_neur()
{
   printf("\nLet's face it, you're \
neurotic.\n\n");
   A_happy_today();
}

/* The State Table is defined as a structure
   of ints and functions                          */


typedef struct
{
   enum T_state  state;
   enum T_event  event;
   void          (*action)();
   enum T_state  next_state;
} STATE_TABLE;

/* Declare a state table and initialise it */
STATE_TABLE          state_table[] =
{
  S_start,     E_yes, A_happy_yest,  S_happy,
  S_start,     E_no,  A_happy_yest,  S_not_happy,
  S_happy,     E_yes, A_conc_content,S_start,
  S_happy,     E_no,  A_conc_neur,   S_start,
  S_not_happy,E_yes, A_conc_neur,   S_start,
```

Figure 4 - The State Table Interpreter Code

```
   S_not_happy,E_no,  A_conc_mis,    S_start,
   S_last,      0,    A_happy_today, 0
};

#define LAST_STATE  6  /* last index of array */


/* The table interpreter functions */


/* The dummy Event Collector simply translates
   user input into the events */

enum T_event wait_for_event()
{
   char  c[4];

   printf("Please enter 'yes' or 'no' : ");
   gets(c);
   if (c[0] == 'y')
      return(E_yes);
   else
      return(E_no);
}


/* The State Table Interpreter itself */

void st_interp()
{
   enum T_state    state;  /* the current state */
   enum T_event    event;  /* the current event */
   STATE_TABLE     *p;     /* the state table   */

   state = S_start;        /* initialise state  */
   A_happy_today();        /* initial action    */

   while(1)                /* do forever        */
   {
      event = wait_for_event();
      for (p = &state_table[0];
           p < &state_table[LAST_STATE];
           p++)
      {
         if ((p->state == state) &&
             (event == p->event))
         {
            (*p->action)();  /* do the action */
            state = p->next_state;
            break;        /* back to outer loop */
         }
      }
   }
}

main()
{
   printf("State table interpreter\n\n");
   st_interp();
}
```

Figure 4 - The State Table Interpreter Code (Continued)

What else Dennis says

In the concluding part of Peter Collinson's interview with Dennis Ritchie, they talk about C, Pascal, FORTRAN and that Johnny-come-lately C++.

An important feature of C is the way it lets you bend the rules. There are always legitimate situations where you need to get around the rules, particularly if you are treating C as an assembler rather than coming at it from the Pascal end of the world.

There are very interesting philosophical differences between what are otherwise very similar languages like Pascal and C. When you look at the genuine operations that Pascal permits, the form of discourse is really essentially analogous to C. The differences are tiny and, of course, some of them are annoying in practice. The genuine philosophical difference is that C was always considered to be a tool for expressing what you wanted to do. Affecting the environment of a program was always very much a part of the use of the language.

By that I mean it was a systems programming language.The point of it was not merely to express algorithms but also to conveniently interface with the rest of the world in a style that we developed. Very early on, all of the system calls were made available. In fact, the system calls were adjusted or in some cases altered so that the data structures passed into the operating system were compatible with C.

Pascal is at the other extreme in the sense that it is a 'teaching language'. Therefore, the custom is that it checks things much more strictly so it can provide students with more safety and feedback. Even more than that, it was much more internally oriented in that it was supposed to be a way of expressing algorithms in which you compute a bunch of numbers and print the answer. Its relations with the outside world were just enough to put the necessary numbers in and get the answer out again; as opposed to controlling the environment. The crucial thing is the inward looking notion of Pascal.

Some of the same things happen in FORTRAN, even though that was always intended to be a practical language. For example, FORTRAN has a model of I/O with files, whether or not it's very good, it's at least consistent internally. The point is that it's defined only in terms of the language itself, the notion of records and sequential access. The concepts and

The differences between C and Pascal are tiny - and some of them are annoying

operations on files are defined in terms of the FORTRAN standard. In any actual implementation of FORTRAN you have the question of saying 'if I really want to talk to a FORTRAN program, and I'm not part of the FORTRAN world, how is their abstraction represented in my real system?' then you look up these great books and look up data layouts, there are counts and all sorts of things, so you find out how to do it.

In C, as embodied in UNIX, that sort of thing just doesn't exist. The data structures and so forth are not hidden as part of the runtime environment, the language was always intended to be independent of the actual environment. The I/O routines are not part of the language, they are shared between the language and the operating system.

I think that when we look back on all this UNIX and C, we will in the end be thinking that it was C that was more important from an historical perspective. It was C that spread out in people's products, rather than UNIX.

It is certainly something that has spread beyond UNIX, but that's a relatively recent phenomenon, it didn't happen until five or six years ago.

Going ANSI

How did you feel when the spread lead to the ANSI standardisation process?

I was very happy to see this happen. it was clear that some sort of much more formal and complete description was needed, but it was a job that I didn't want to do. It was intolerable that C was being used so much, but there was no formal specification of the language. It was very lucky that whatever there was in the sociology of the situation had made the language as beautiful as it was throughout the various phases. There were little differences in people's compilers, but it was surprising how few dialects there were.

Wasn't that a result of your conscious decision to put the language into the White Book?

No, it's not surprising that most people conformed to that. The problem was that there had been extensions beyond that. People did add things, and people didn't do all things in quite the same way. There were things like `far` and `near` pointers, put in by desperation by the 80X86 people. There were mistaken implementations in which people consciously did things that they shouldn't have. But compared to Pascal, the number of important, different dialects were few, and there was very little variation among the different dialects.

Even independent of that, it was clear that having an agreed upon definition was very

important. I didn't want to do it. It would have been a lot of work because it was already obvious even by the time ANSI started that there was a fairly hefty political angle to the whole thing. I could no longer be autocratic and say 'this is the way things are'.

Doug McIlroy was the one who started pushing things towards an ANSI standard, and Larry Rosler was the one who did much of the work. I am really quite happy about the way things have worked out. In terms of the standards process, the committee was quite a good model.

The ANSI committee had to clear up definitions, formalise the rules and take care of all the things that had been added to the language since the White Book. It was very clear that the biggest weakness was this area of the 'type of a function'. This was not properly taken care of in the type structure, and the type of a function should include the types of its arguments. For at least two or three years before ANSI and also before C++ or C-with-classes, it was clear that something like this was needed based on discussions at some internal meetings inside the Labs about C standards.

Standard compromise

When ANSI began to formulate the notation, it didn't come out of nowhere, they took the ideas from C-with-classes. It was clear to me that this was a real two edged thing; and that there were three ways that they could go. One is to say that 'we're sorry but things really should have been done differently, but it's too late, forget about it'.

The other extreme is to say: the language has now changed, function parameters *must* be given, templates *must* be present giving types of function parameters; otherwise the program will not compile. Either you write in old fashioned C or ANSI C. The third, intermediate position is that you should mix both approaches, making templates optional.

I was worried about the middle position and wrote a letter objecting to it. It was obvious to me that this wasn't the way to do it. The reason was that you're in this sort of limbo, a transition period where various options are possible. It confuses the new people learning the language because there are two ways of doing things. It has the potential for very subtle bugs in that you can write a call under the impression that you are calling a function that's been prototyped when, in fact, it's not and so the conversion sequence that you are expecting to be done just doesn't happen.

In general, it seemed to be an undesirable situation to have this half way state that was going to last indefinitely. Nevertheless, I wasn't surprised when the ANSI committee said that it's not possible to declare a flag day and everyone switch to the new compiler; and yet we need the facility so much that we cannot just pass it by. Almost all the other changes that they made were designed in such a way that the transition from the old to the new could be done smoothly. This one was bound to be hard to do.

The Plan 9 stuff is exclusively written in ANSI C using a compiler done by Ken Thompson

There are still some problems with what has been defined. For example, in the templates, you can give names that are ignored. You can say

```
extern int f(int arg);
```

The `arg` goes away, it's almost a comment. Yet it can be captured by macro expansion in the pre-processor. This can be a problem, but the name in the template can give you information, like:

```
extern char *strcpy(char *old,
                    char *new);
```

where the parameter names give you some help.

The point is that I don't really disagree with what ANSI did as far as the function prototypes were concerned because it was the only approach that was politically possible and there were strong technical reasons for having the prototypes.

As far as the other things are concerned. It has seemed a good idea to standardise the library, to create a standard list of supported functions rather than relying on using an emulation of some subset of the UNIX library. Most of the UNIX library was designed in a way that it could be used on other systems. One could say, and Ken Thompson and Rob Pike do, that ANSI went too far and improved too many things. I think that you can make that argument, but I guess that doesn't bother me too much.

There are doubtless many technical areas where they could have done better. But taking a step back, I think the standard was really needed for a lot of reasons. The biggest benefit is that a coherent and complete definition of the language exists. Second, a lot of the excessive informality of the original White Book could not be got rid of without making an actual formal definition. Third, the language was becoming used enough that the political desirability of having a national and then an international standard was growing greater. There were places where it would not be used unless it were blessed by some sort of international body; the language needed that official seal of approval.

I guess I used this phrase at the UKUUG meeting in London: they took something that was basically not broken, but in serious need of clarification and updating, and clarified it and updated it without breaking it.

What are you doing about ANSI C internally in your group?

First, the Plan 9 stuff is exclusively written in ANSI C using a compiler done by Ken Thompson. The compiler insists that prototypes be used. We are using this as an opportunity to clean up the code. We are not just mechanically translating the old code but we are taking the chance to look at things. It's a new style ANSI C compiler that is not backwards compatible.

Second, the various people are contributing to what we call an ANSI/POSIX environment, where we try to get an environment that supports the ANSI/POSIX world running under our non-POSIX compatible UNIX. We want to have a library and an environment such that we can wheel in programs that conform to ANSI C and POSIX and compile them without any fuss; without having to fiddle with header files and libraries.

C++

How do you feel about C++?

Well, Bjarne and I have divided up the world, he has C++ and I have C, and we don't trample on each other's areas. I have not used C++ 'for real'.

C++ has a number of inherited problems, not from the language point of view but from what it tries to do. It has a real tension because it derives somewhat from C and for the most part tries to leave C alone, in that C programs are by and large supposed to be C++ programs. Of course, there are genuine exceptions to this. C++ is carrying

CIRCLE NO. 614

along a lot of baggage both syntactic and semantic, there are syntactic restrictions that C imposes on the parsing, they can't do the things that they want to do to parse forms of the syntax.

There are a bunch of almost duplications of features, that are left over for historical reasons. For example, do you really need both 'references' and 'pointers'? C chose to make indirection very explicit by having an operator to do it. Whereas ALGOL 68 chose to make it implicit by having it built into the mode and specifying coercion rules, so there was no indirection operator, it just simply happened using references. Now C++ has both types of indirection and has not made a clear choice. Maybe it should.

One can observe that if they had been able to start afresh and design it from scratch, it would be much less constrained. At the same time it was clear why they chose to use C to support C++, they got an enormous amount of value out of using the similarities. First, programs can be easily migrated from C to C++. There is a lot of familiarity. You can approach C++ by saying that this is just like C and then work into the things that are far more distinctive about the object oriented approach. Second, they bootstrapped heavily and successfully from the existing technology by having C++ be typically implemented on top of C.

Bjarne and I have divided up the world. He has C++ and I have C

There are more compilers now, of course, but that's not really necessary. It's clear that if you have control of your environment there's a lot of advantages in doing a compiler in terms of efficiency, debugability etc etc. On the other hand, if you want to move it to a new machine or environment, then an interpreter running on C is a very successful technique. In fact, this is not unique to C++, I think that all the Modula 3 stuff at DEC translates into C - but I am not positive on that. It's a very common technique and a lot of people are doing this, using C literally as an assembler.

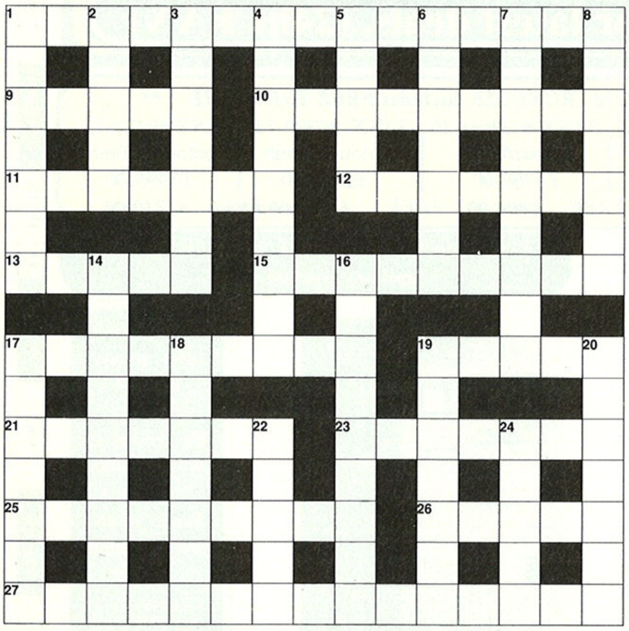
We've almost come full circle, with C as a high-level assembler.

The penalty that is paid for writing in a high-level language certainly exists. There were stories of people who had ported their spreadsheet or editors from assembler to C and totally failed to have something that was commercially viable. There are circumstances where efficiency has been a real problem. A long time ago, I think it was in the original paper describing the portability that Steve Johnson and I wrote: we said that we don't even know what the penalty is for writing in C; and we don't care either. At some level it just doesn't matter. This is good enough and the benefits that we are getting so far outweigh any possible speed-up that you could get by using assembler - that you just don't care. It might be nice to know and study to make things faster, but that does not affect the decision to use C. EXE

Many thanks to Dennis Ritchie for generously taking the time to give this interview.

Peter Collinson is a freelance consultant specialising in UNIX. He can be reached as pc@hillside.co.uk *electronically (although your mailer might be happier to put the address the other way round) or by phone on 0227 761824.*

.EXEWORD MARCH



ACROSS

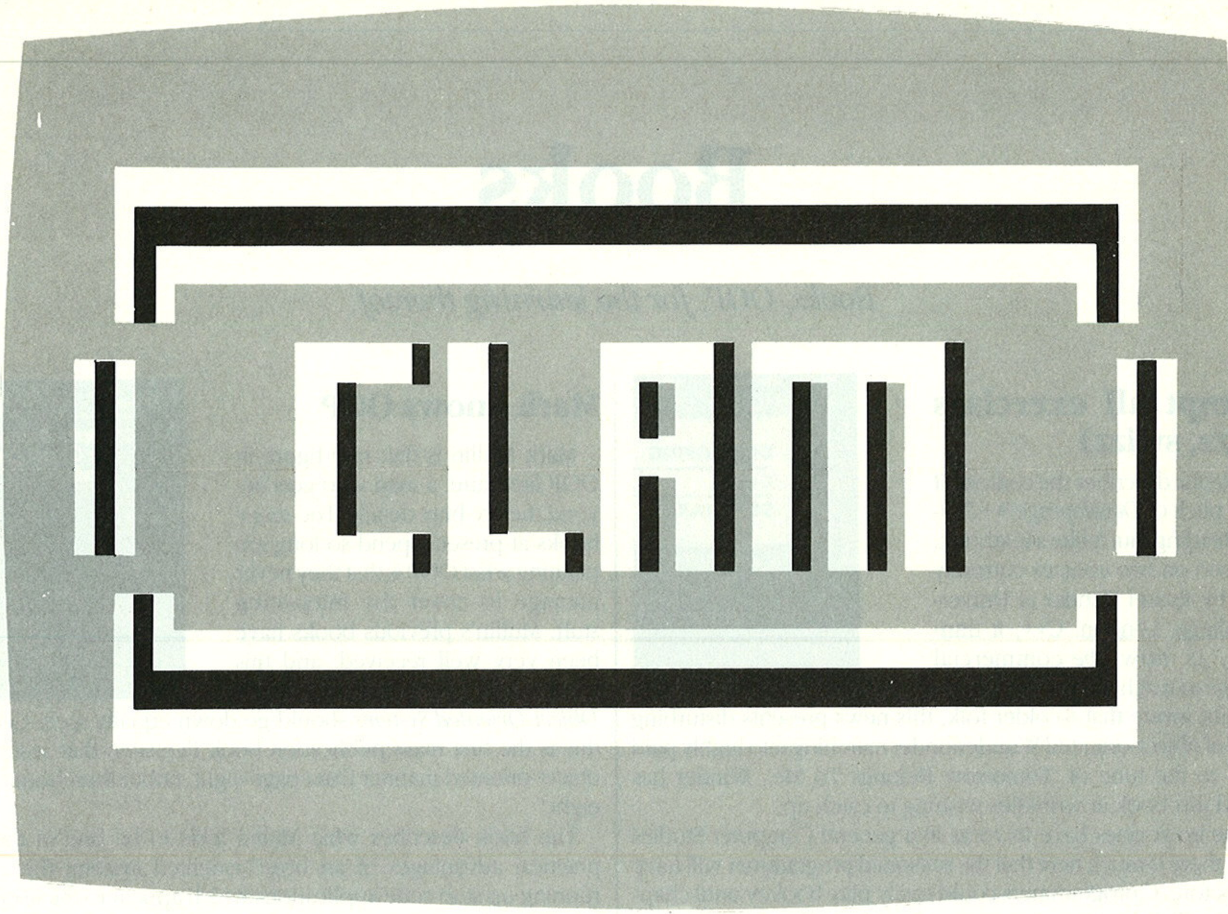
1 You can enter at once with this (9, 6)
9 I call round for purple girl...(5)
10 ...able to teach as Stalin Inc maybe (3, 6)
11 Rubbish! It's G for short! (7)
12 Someone from mid-East is real, I wonder? (7)
13 Maybe it's filthy, but I like it...(5)
15 ...singing wildly on high (9)
17 Seen, I send round (though in a state of poverty) (9)
19 Bill may open them and/or...(5)
21 ...find the y-variable (7)
23 Loader to hold things together...(7)
25 ...in an awful fashion (only he is U) (9)
26 From bottom to top may take to the hills (5)
27 Looking for 13, where programmers hope to enter (7,8)

DOWN

1 It's wrong to be sick, even in France (7)
2 Base of storage on tape or floppy (5)
3 State your 21s at the start (7)
4 Initial model for the printer? (9)
5 Unusual boredom from rotten nuisance (5)
6 Not a distributing processor! (7)
7 Crazy type who likes the ends of 26 (9)
8 Capitalism's main job maybe (7)
14 Hold back parts used in parity (5, 4)
16 Show on? Keep showing! (7, 2)
17 Found with ones or crosses...(7)
18 ...nuclear species of poet, so I change (7)
19 Such high ranking knowledge may be global (7)
20 Old group involved in searches (7)
22 Homely places for looping the loop? (5)
24 Friendly relations when a nut's cased (5)

H	I	G	H	W	A	Y	S		T	R	A	C	E	D
E		A		A		E				E		A		E
A	C	T	I	V	I	T	Y		L	A	R	R	U	P
D		H		E		I		S		D		A		L
E	V	E	N	L	Y		B	Y	T	E	W	I	S	E
R		R		E		P		S		R		A		T
			U	T	I	L	I	T	Y		O	G	R	E
A		I		S		O		E		A		E		D
D	A	T	A		I	T	E	M	I	S	E			
A		E		F		T		S		S		V		M
P	A	R	A	L	L	E	L		D	E	V	I	C	E
T		A		O		R		M		M		S		S
I	N	T	E	R	N		A	I	R	B	R	U	S	H
V		E		I				C		L		A		E
E	P	S	O	N	S		R	E	C	Y	C	L	E	S

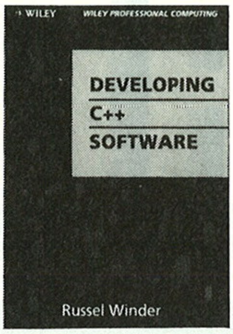
.EXEWORD FEBRUARY


CLAM

Books

Books, OOP, for the learning thereof.

Attempt all exercises (swizz, swizz)



Academic describes the content, if not the pitch of *Developing C++ Software.* Reading much like a textbook, it is based on two courses currently taught by Russel Winder at University College, London. C++, it transpires, is now *the* commercial language to teach the young sprogs. No doubt aware that to older folk, this news presents disturbing visions of object-oriented Wunderkinder marching into highly paid careers to the tune of 'Tomorrow Belongs To Me', Winder has targeted this book at wrinklies wishing to catch up.

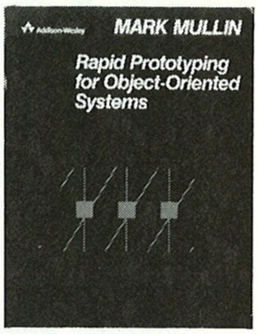
As the book does have its roots in a general Computer Studies course, there is much here that the seasoned programmer will have heard before. C programmers could easily play hookey until chapter eight. Slightly more surprising, though, is the initial lack of attention to the creeds of object-orientation. Winder's early treatment is stoically traditional, with procedures and data structures at arm's length from one another. Despite his disparaging of C as a beginner's language, surprisingly little time is spent throughout on the features that set C++ apart. This is an effect, I think, of C++'s ancestry rather than Winder's design. The bolt-on nature of classes into the C language means that to understand their syntax, one really needs to understand C. And to understand C, one has to go through all that 'how signed numbers are implemented' palaver. Winder doesn't like to admit this, which is why he is sometimes forced to write pedagogic dodges similar to 'put a \ at the end of each line of your generic class definition and don't ask why'. A good move in a Computer Science course; possibly more annoying to the curious reader.

Nevertheless, when the shores of OOPs are finally sighted, Winder's explorations are exhaustive and informative. It is intended to be a sit-down-and-type tutorial, too, so there are lots (and lots) of examples.

In summary then, this is a book which errs on the side of inclusion. It includes asides on C and traditional programming methods, it includes examinations of C++ compiler quirks including a few decidedly unacademic shortcuts. Non-C, but still accomplished, programmers can quickly get up to speed with it: C users playing catch-up will have bought a quick and readable update to their knowledge.

Title: Developing C++ Software — *Pages: 400*
Author: Russel Winder — *Price: £20.65*
Publisher: Wiley — *ISBN: 0-471-923848*

Mark knows OOP



Mark Mullin is that rare figure in OOP literature: a man who gets beyond the itty-bitty details. Too many books at present spend so long explaining what OOP is that they never manage to cover the interesting stuff. Mullin's previous books have been very well received, and this new volume, *Rapid Prototyping for Object-Oriented Systems* should go down equally well. Certainly, this is the first mass publication book I've seen that assumes an object-oriented manner from page eight, rather than from chapter eight.

The book describes what Mullin feels to be one of the great practical advantages of an object-oriented system; the ease of running up stub code applications which appear to the user just as a final product will appear. Demos, in other words. Users of, say, Common View or Turbo Vision will understand his point. The viability, with these libraries, of sketching an entire interface within a few lines of code is one of the more pleasant of OOP phenomena.

Mullin has taken this ability, and made it a central part of the development process. As soon as a vague description is given, he argues, the programmer should demonstrate a working prototype of his program. Any specification ambiguities can then be ironed in a week or so of feedback between designer and user. Then, once all has been smoothed out, the programmer will be 'left in the enviable position of working, not from a written or formal description, but with a viable outline of his program's internal structure.

It sounds slightly pie-in-the-sky, but Mullin has a great deal of experience with OOP systems, and demonstrates the technique with realistic examples. He also includes telling pieces on topics such as how to stop the client from meddling too much, which shows some down-to-earth gumption in his idealism.

Still, not everyone loves Mullin's style. As he himself admits, he is not a 'cookbook author', and prefers crafting metaphors rather than solid code examples. But Mullin does write to some point, and, most important, he knows his stuff. In fact, the only minor criticism I have of this book is that what example code there is, is in Smalltalk, a language few people can claim to know. Even so, I expect OOPers of all creeds to have no problems understanding and enjoying this informative book.

Title: Rapid Prototyping for Object-Oriented Systems — *Pages: 220*
Author: Mark Mullin — *Price: £20.65*
Publisher: Addison-Wesley — *ISBN: 0-201-55024-5*

Books Received This Month

Thinking In Postscript by Glenn C Reid	Addison-Wesley	£20.65	ISBN 0-201 52372-8	pp207
Operating System Concepts (3rd Edition) by Scilberschatz, Peterson, Galvin	Addison-Wesley	£21.95	ISBN 0-201 51379X	pp694
Power of Paradox by Francis Botto	Sigma	£16.95	ISBN 1-85058-208	pp215
X Window Systems Toolkit by P J Asente, and Ralph R Swick	Digital Press	£23.95	ISBN 1-55558-051-3	pp967
Microsoft Quickbasic Bible V4.5 by The Waite Group	Microsoft Press	£24.95	ISBN 1-55615-262-0	pp935

Editorial Index

If you are interested in the News articles on pages 4, 5 & 6, and would like to receive more information about the products mentioned, please circle the corresponding circle number on the Reader Service Card at the back of this magazine.

COMPANY	PRODUCT	CIRCLE NO
American Cybernetics	*Multi-Edit*	621
Avantek	*Access Mac*	622
Borland	*Sidekick*	623
Chorus Systeme	*UNIX*	624
Ctrl Alt Deli	*HotShot*	625
Equisys	*Isys*	626
Gimpel	*C-terp*	627
IMS	*DOS 386 Professional*	628
Intel	*DOS/RMX*	629
Lotus	*DataLens Driver*	630
MIPS Computer Systems	*R4000*	631
Performance Technologies	*Disk Duplicator Plus*	632
Phar Lap	*286 DOS Extender*	633
Precision Software	*Superbase 4*	634
Sage	*Demo II*	635
Santa Cruz Operation	*Developer Conference*	636
Sequiter	*Codebase ++*	637
Symantec	*Text Editor*	638
System Science	*Programmer's Toolbox*	639
Turbo Power	*TSR Made Easy*	640
UNIX International	*UNIX API*	641

.EXE

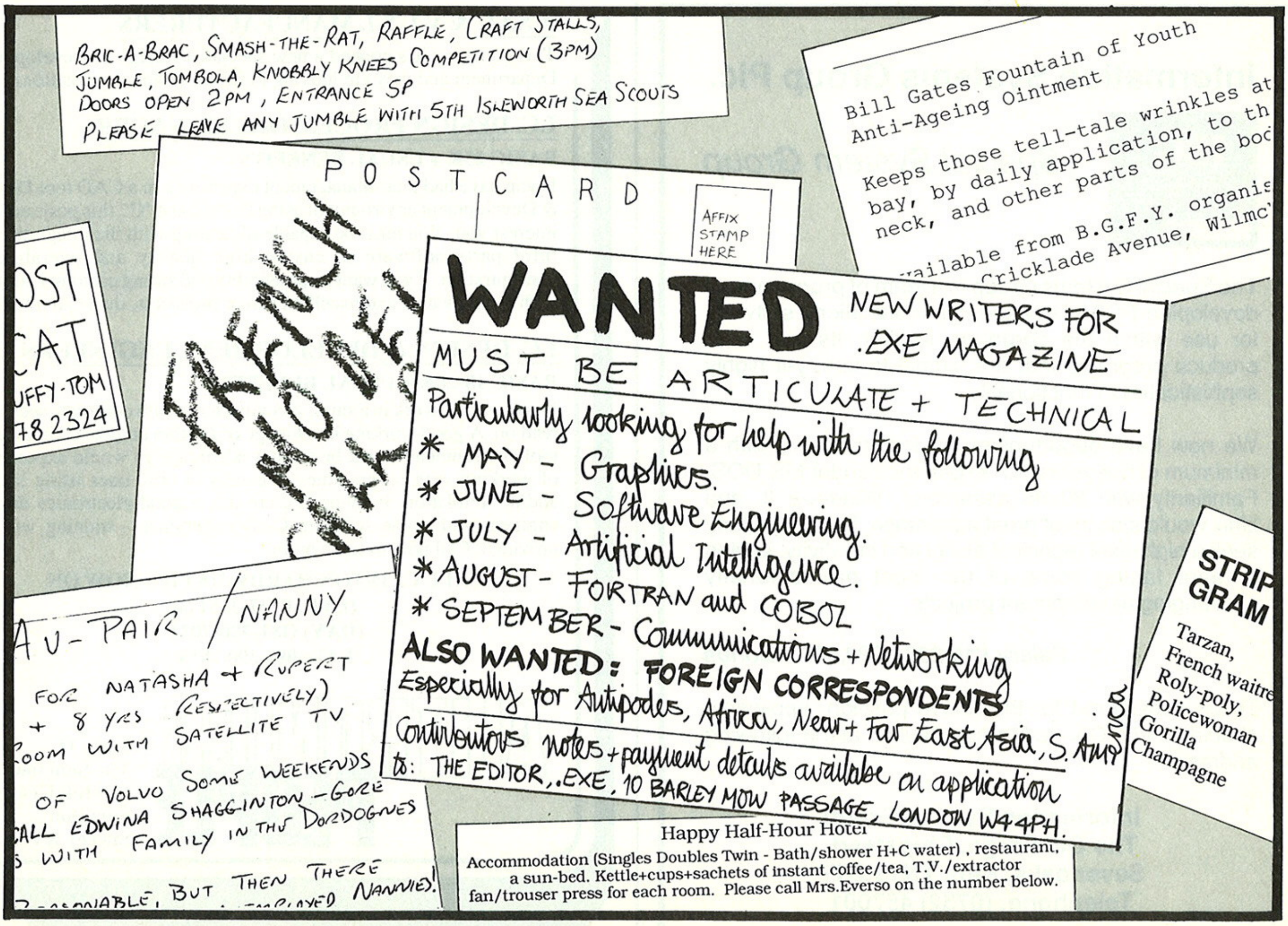
The Next Quarter

April — **Life after DOS.** As machines grow more powerful, MS-DOS seems set to fall behind. But what will replace it? Windows? We also investigate how to write programs which can cope with any operating system. Plus The Software Training Guide.

May — **Graphics.** Incorporating graphics into programs is rewarding, but is often fraught with difficulties. We look at some advanced techniques, and review some of the utilities that can help.

June — **Software Tools Show Special.** **The Software Engineers.** Modern employers don't advertise for 'programmers', they ask for 'software engineers'. But what is a software engineer, and what are the tools and techniques at his disposal?

July — **Artificial Intelligence.** AI, according to its proponents, is coming out of the labss and into everyday use. We will be cutting through the mystique associated with thiss field, and finding out just what modern AI is capable of.



ADVERTISERS INDEX

ADVERTISER	PRODUCT/SERVICE	CIRCLE	PAGE
7X Computers	UNIX Development Systems	611	80
.EXE Directory	.EXE Selection of Goodies	617	90
Addison Wesley	Book Publishers	616	89
A.L. Downloading	Disk Conversion	610	79
Biton	Clipper Library	612	81
Bits Per Second	Graphics Server Software	582	52
BL Security	Software Protection	598	71
Blackwell Scientific	Windows BASIC Compiler	574	43
Blenheim Online	Exhibition	593	67
Blinker	Clipper Fast Linker	590	62
Brent Communications	MAX copy PROtection	597	71
Broadchain	Disk Distributor	605	76
Camel	Plotter Utilities	599	73
CEBRA	Multi VGA Adapters	585	55
Clearsoft/Microphar	Software Protection	618	OBC
CNS/Zortech	MS Window Library	568	31
CTL	Copy Protection Hardware	600	73
Dataflex	Applications Database	575	45
DES	Software Protection	592	65
Diad/Zortech	Array extension Library	567	29
Deepak Sareen	Project Management Software	596	70
Fox Software	Fox Developers Conference	569	32
Grey Matter	Programming Tools	551	3
Intasoft	Software Management System	554	11
Jensen & Partners	C++ Compiler	562	23
Jensen & Partners	Competition	-	25
Kibworth	C, C++, Pascal Training	564	25
KJD	IBM PC Upgrade	594	68
Lahey	FORTRAN Compilers	603	75
LBMS	Multi-User CASE	576	46
Linx I	All UNIX Systems	604	75
Linx II	All UNIX Systems	614	85
Logic Programming Ass	AI/KBS/OOPS Software	579	51
Magnifeye	Software Protection Device	571	37
Metaglyphix	X Windows for DOS	586	55

ADVERTISER	PRODUCT/SERVICE	CIRCLE	PAGE
Microcosm	Copy Protection Software	606	76
Microft Technology	Network Security	615	87
Microsoft	Windows Development	555	15
Microsoft	Windows Development	557	17
MSE	Applications Generator	611	80
Netware	Networking	595	69
Nu-Mega	Debugging Tools	558	18
OTG Systems	Marketing Consultancy	563	79
Pecan Software	The Pecan Power System	559	21
Pinna Electronics I	Graphics Menu	556	17
Pinna Electronics II	Tools & Graphics	607	77
Polyhedron	FORTRAN Compiler	602	75
Prospero	DOS Extender Kit	561	22
QA I	C Programming Training	550	IFC
QAII	C++ Windows 3 Library	588	59
QBS	Clipper Add-ons	578	50
Rainbow Technologies	Software Data Security	552	7
Recital	RDBMS/4GL for VAX & UNIX	584	54
SCL	Communications Boards	580	51
Select Software	CASE Tools	565	25
Softlok International	Piracy Protection	609	78
S/ware Con. Co List	Development Tools	577	49
S/ware Generation I	Editor	587	57
S/ware Generation II	Code Generator	570	35
S/ware Paradise	Business Software Sales	589	60/61
Solution Systems	Programming Editor	573	41
System Science I	Windows Tools	601	73
System Science II	Development Tools	608	77
USA Software	Programming Tools	566	26
User Friendly	Software Copy Protection	560	21
Vleermuis	C++ Class Tree OS/2	553	8/9
Vogon	Data Interchange Software	581	51
Zortech I	Multi Plat C++ Comps	572	38
Zortech II	C++ Video Tutorial	583	53

STOB - Underground Liff

'The Meaning of Liff' is a book by Douglas Adams and John Lloyd, in which useful words are coined from the names of towns. Stob the plagiarist has applied the technique to programming terms, using the London Underground as her starting point.

bank *n.* A bank of memory is a block of RAM which can be switched in and out of the processor's address space. Gosh, that was easy, wasn't it? I wish there were a few more like this. Perhaps London Transport could open a Virtual Base Class East station on the proposed Jubilee extension, and we'd be through in a minute.

Dollis hill *n.* Named after Professor Esther Dollis of the Computer and Domestic Science Department, University of Chiswick and Fashionable West London. When on the track of a particularly elusive bug, a programmer, if he is not careful, will start to climb a Dollis hill. First he will blame his compiler, which must be producing incorrect code; then he will become convinced that there is a mistake in the library; next he will wonder if something is wrong with his machine and so on. The peak of a Dollis hill is to suspect a fault in the electric main. Needless to say, 87% of Dollis hills are caused by operations on unassigned pointers.

fairlop *n.* A fairlop is an annoying pause or delay, caused by circumstances outside the programmer's control (but often blamed on him). A real-time animation that pauses every few seconds for a garbage collect is said to 'burp a fairlop'. The hourglass 'please wait' icon used by GUIs signals a blatant fairlop.

goodge street *n.* Those of you who spotted that 'oo' will have already dived for cover to dodge the incoming acronym. 'Goodge' stands for Gnomic Object-Oriented Designers Get Everywhere; if you are down the pub and everybody around you is saying things like 'Windows programming is *so* much easier with a few C++ classes', then you are definitely on GOODGE Street.

holborn *n.* A holborn is a mistake in a work of hi-tech fiction which indicates the depth of research undertaken by the author. A classic example may be found in a Dick Francis novel about a 'get-rich-quick' computer program that predicts the winners of horse races. We are only vouchsafed a few lines of this wonderful BASIC program, but one of these goes something like this:

```
560 IF I < P THEN 730: I5 = 40
```

We trust `I5` is nothing important. A good episode of *Dr Who* may contain as many as five distinct holborns.

loughton *v.* To loughton is to attempt to create a system using manifestly unsuitable tools. Trying to write a C compiler in dBASE language, or a simple database in assembly language are examples of loughtoning. Loughtoners are typically individuals who are fanatical about the system that they use, or are just stupid, or both.

pimlico *a.* An ornate style of program code commenting, probably incorporating many sloane squares (*qv*). If it takes less time to fix a bug in a routine than it does to alter the accompanying comment, then your coding is definitely too pimlico.

sloane square *n.* A program comment contained entirely in a box of *******s. A leicester square, incidentally, is a bastard sloane square where all the verticals are out of alignment as a result, for example, of the printer having the wrong tab value.

stepney green *n.* A programmer who expects his code to compile first time.

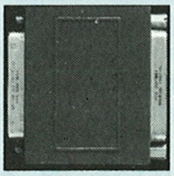
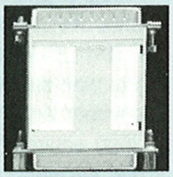
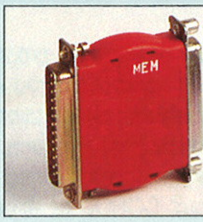
turnham green *n.* A specialised graphics-fill routine which enhances the verdure of screen entities. Wow.

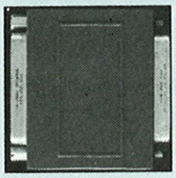
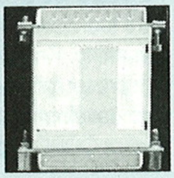
virtual base class east *n.* Just seeing if you were still awake.

EXE

SOFTWARE PROTECTION!

Can You Spot The Difference?

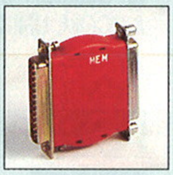



- ☑ High-level security keys
- ☑ Assembler-based, customer specific, encrypted interrogation routines
- ☑ Over 70 languages supported
- ☑ Compatibility - due to 8 years experience & 600,000+ keys sold
- ☑ Reliable on-going support
- ☑ Physically unique keys for each customer
- ☑ MS-DOS, OS/2, UNIX, XENIX, WINDOWS 2 & 3, "MACINTOSH"
- ☑ Transparent operation with most peripherals on market
- ☑ Free language updates
- ☑ Parallel, Serial, Mac ports

OVER 3,100 CUSTOMERS WORLDWIDE HAVE!

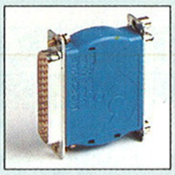
Electronic Key

The ideal device for identically produced software packages. Uniquely wired with customer code and a software code. Uses Assembler based program, decryption interface and random values.



1 Word Memory Key

Custom hardware wiring allows the developer total control over information stored in the key. 2 bytes of memory allows several packages to be protected with just the one key.



31 Word Memory Key

For multiple software protection schemes. 31 words of 16 bits of non-volatile dynamically programmable memory. Its capacity to store information provides virtually limitless power. Flexible protection scheme can be modified on-site during operation of software package.

Micro Processor Key

Provides the ultimate in software security. Not tied to any language or O/S. 8 bit microprocessor powers from RS-232 level. Requires no power supply. For PC terminals, minis, & others using RS 232 C comms. Used on workstations. This key is effectively a computer.

Macintosh Key

Extremely powerful & customised protection for the Mac. 31 words available for random storage. Providing unequalled protection the Macintosh Memory Key connects to the SCSI 25 pin port and operates transparently.

MICROPHAR... The European Standard!

For a no-nonsense informative discussion on how our dongles can better protect your profits, please call us on:

091-378 9191

CLE

CLEARSOFT, L

Langley Moor, I

Telephone: (09

Please send me full details on your protection systems.

Name: ....

Position: ....

Address: ....